TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Steuerungs- und Regelungstechnik

Design and Control of Shape Rendering Interfaces

Stefan Klare

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr.-Ing. habil. Gerhard Rigoll

Prüfer der Dissertation:

1. TUM-IAS Junior Fellow Dr.-Ing. Angelika Peer
2. Univ.-Prof. Gordon Cheng, Ph.D.

Die Dissertation wurde am 16.01.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 30.09.2014 angenommen.

Foreword

This thesis summarizes four years of my research carried out at the Institute of Automatic Control Engineering (LSR), Technische Universität München. The work is supported in part by the BEAMING project within the 7th Framework Programme of the European Union.
First of all, I would like to thank my thesis advisor Dr.-Ing Angelika Peer, who guided me through academia to where I am today. She always left me enough space and freedom for exploration and innovation.
My next most sincere appreciation goes to Prof. Dr.-Ing/Univ. Tokio Martin Buss who offered me a research position in one of the best-established robotics group.
Thanks go to the technicians, Mr. Gradl, Mr. Kubick, Mr. Geng, Mr. Stöber, Mr. Lowitz, Mr. Jaschik and especially Mr. Weilbach who always helped me to realize my ideas.
Next, I would like to thank Dr. Dirk Wollherr and Dr. Marion Leibold as well as the secretaries Ms. Werner, Ms. Schmid and Ms. Renner who had always kindly helped me with administrative issues.
Further thanks goes to my colleagues who inspired me and helped with intensive discussions. I especially want to thank Markus Rank, Laith Alkurdi, Sheraz Khan, Alexander Schmidts, Daniel Carton, Roderick de Nijs, Ken Friedl, Andreas Lawitzky, Sebastian Söntges, Stefan Sosnowski, Harald Voit, Moritz Stötter, Thomas Schauss, Daniel Althoff, and everybody I forgot. I had a lot of fun with all of you.
A very special thank goes to my friends who always supported me during my thesis. Especially I want to thank Carlo von Reumont, Dr. Nikolai von Hoyningen-Hüne, Philipp Scherer, Andreas Massat, Till Neßeler, Andreas Trapp and Lukas Stubenrauch.
My final and most sincere thanks go to my family, who had always put their faith in me.

Munich, December 2013 Stefan Klare

to my parents

Abstract

This thesis shows a systematic analysis for the design and control of 3D shape rendering interfaces (SRI). The corresponding research field is still very young and no commercial solutions for SRIs are existent up to now. Scientific solutions for the design of SRIs are not satisfying as they have a low spatial resolution, are not feasible to display objects with undercuts or are not actuated. Plenty of promising applications for SRIs are thinkable. Potential applications lie in the fields of telerobotics, virtual reality and design. A SRI, which is used in the design process of e.g. car bodies, on the one hand offers the opportunity to display the shape of specific parts which were previously designed by e.g. CAD-software tools. On the other hand, changes to these parts can be made intuitively by interacting with a SRI. Other promising applications can be found in the fields of virtual or augmented reality. As haptic interactions of a human operator with the SRI occurs with the bare hand, a high transparency can be gained for the system. Thereby, the combination of SRIs with encounter-type interfaces is of special interest, as the workspace of SRIs can be extended by the encounter-type interface.
This dissertation describes a systematic approach for the design of SRIs. After defining design guidelines, a numerical method to quantitatively compare different SRIs or different configurations of it, is presented. This method is used to gain an optimal hardware design to render cylindrical and spherical objects. The derived design, a parallel kinematics with 24 Degrees of freedom (DoF), is implemented in a prototype. The prototype has a higher resolution than the existing analogues, is fully actuated and can be extended easily. Maximum curvatures, stiffness and dynamics are determined for the prototype.
Further, a kinematic description for SRIs, that are characterized by a high amount of DoF and a parallel kinematic design, is presented. Existing methods, which are based on numerical optimizations are not real-time capable. In contrast, the method presented here is real-time capable as the kinematics is solved in a differential way. For this purpose, tasks are introduced, which must be satisfied simultaneously. The tasks are embedded in a hierarchical framework in accordance to their importance, and take care of the kinematic constraints of the parallel kinematics, the shape to render and user-interactions. The kinematic description is divided into two modes. For haptic interaction with compliant objects a different description is used as for interactions with stiff objects. Furthermore, two shape descriptions can be distinguished. The shape (surface) of objects can either be described by implicit functions or with the help of polygon data sets, which is a prevalent shape description in common CAD software tools.
The ambition to cover a large interaction area with a kinematics which features a high resolution inevitably leads to a kinematics with a high amount of actuators, which further leads to an expensive hardware design. Thus, this thesis analyses a measure, which can be used to reduce the amount of needed actuators for SRIs. Sequentially actuating the joints with one single actuator only is simulated and a method for determining the time-optimal switching sequence is analyzed. For this, a hybrid system description is derived. The optimization problem is solved with a heuristic graph search method.
Thus, this thesis gives extensive and innovative directions for the design and control of SRIs, which exceed state-of-the-art approaches.

Zusammenfassung

In der vorliegenden Dissertationsschrift wird eine umfangreiche und systematische Analyse für das Design und die Regelung von 3D-Formdarstellungsgeräten (SRI) vorgestellt. Der entsprechende Forschungsbereich ist noch sehr jung und derzeit sind keine kommerziellen Lösungen für SRIs bekannt. Wissenschaftliche Lösungen für das Design von SRIs sind unbefriedigend, da sie eine geringe räumliche Auflösung besitzen, keine Hinterschnitte darstellen können, oder nicht aktuiert sind. Vielversprechende Anwendungsbereiche für SRIs sind naheliegend. Mögliche Anwendungen liegen in den Bereichen der Telerobotik, der virtuellen Realität und im Design. Ein SRI, das beispielsweise im Design-Prozess von Fahrzeugkarosserieteilen verwendet wird, bietet einerseits die Möglichkeit, Formen bestimmter Teile, die beispielsweise mit einer CAD-Software designed wurden zu visualisieren. Andererseits können Änderungen an diesen Teilen in intuitiver Weise vorgenommen werden, indem ein SRI als Eingabegerät verwendet wird. Andere vielversprechende Anwendungen liegen im Bereich der virtuellen oder erweiterten Realität. Da haptische Interaktionen zwischen einem menschlichen Bediener und dem SRI mit der bloßen Hand durchgeführt werden, kann eine hohe Transparenz für das haptische System gewonnen werden. Dabei ist insbesondere die Kombination von SRIs mit Begegnungseingabegeräten (Encounter-type) von Interesse, da der Arbeitsbereich von SRIs durch Begegnungseingabegeräte erweitert werden kann.
Diese Schrift beschreibt ein systematisches Vorgehen für das Design von 3D-Eingabegeräten. Hierzu werden zunächst Designrichtlinien festgelegt. Eine numerische Methode, zum quantitativen Vergleich von verschiedenen Eingabegeräten bzw. verschiedenen Konfigurationen, wird vorgestellt. Diese Methode wird verwendet, um ein optimales Hardwaredesign zur Darstellung von zylindrischen sowie kugelförmigen Objekten zu ermitteln. Das so ermittelte Design, eine Parallelkinematik mit 24 Freiheitsgraden, wird in einen Prototyp umgesetzt. Der Prototyp weißt eine höhere Auflösung als bekannte Alternativlösungen auf, ist voll aktuiert und kann einfach erweitert werden. Maximale Krümmung, Steifigkeit und Dynamik werden für den Prototypen ermittelt.
Zudem wird eine kinematische Beschreibungsweise für Formeingabegeräte präsentiert, die sich durch eine hohe Anzahl von Freiheitsgraden und ein parallelkinematisches Design auszeichnen. Bestehende Verfahren, die auf numerischen Optimierungen basieren, sind nicht echtzeitfähig. Im Gegensatz dazu ist das hier vorgestellte Verfahren echtzeitfähig, da die Kinematik in differenzieller Weise gelöst wird. Hierzu werden sogenannte Teilaufgaben eingeführt, die simultan erfüllt sein müssen. Diese Teilaufgaben werden in eine hierarchische Struktur eingebettet, um deren Wichtigkeit zu berücksichtigen. Die Aufgaben berücksichtigen insbesondere Nebenbedingungen, hervorgerufen durch die geschlossenen kinematischen Ketten des Mechanismus, die darzustellende Form und Interaktionen mit dem Benutzer. Die kinematische Beschreibungsweise wird in zwei Modi unterteilt. Für haptische Interaktionen mit nachgiebigen Objekten wird eine andere Beschreibungsweise verwendet als für Interaktionen mit steifen Objekten. Außerdem werden zwei Formbeschreibungen unterschieden. So kann die Form (Oberfläche) von Objekten entweder als implizite Funktion oder als Polygondatensatz beschrieben werden, was eine verbreitete Beschreibungsweise in den meisten gängigen CAD-Software-Tools ist.
Das Streben nach einer großen Interaktionsfläche der Kinematik mit einer hohen räumlichen

Auflösung führt zwangsläufig zu einer Kinematik mit einer hohen Anzahl an Aktoren, was wiederum zu einem teuren Hardware-Design führt. Daher wird in dieser Arbeit eine Maßnahme diskutiert, die zu einer Reduktion der Anzahl der benötigten Aktuatoren führen sollen. Es wird das sequenzielle Aktuieren von Gelenken mit einem einzigen Aktuator simuliert und eine Vorgehensweise zum Ermitteln der zeitoptimalen Schaltsequenz analysiert. Hierzu wird eine hybride Systembeschreibung hergeleitet. Die Optimierungsaufgabe wird mit einer heuristischen Graphensuche gelöst.
Diese Arbeit bietet also eine umfangreiche und innovative Anleitung für das Design und die Regelung von SRIs, die den Stand der Technik in vielen Gesichtspunkten übertrifft.

Contents

Notations

Abbreviations

FO	Formable Object
DoF	Degrees of Freedom
SFM	Surface Freedom Measure
FGM	Freedom from Ground Measure
DR	Data Resolution
FM	Formability Measure
CAD	Computer Aided Design
ABS	Acrylonitrile Butadiene Styrene
PID	Proportional-Integral-Differential
SFI	Shape Forming Index
MEMS	Micro Electro Mechanical System
FEM	Finite Element Method
STL	Surface Tesselation Language

Conventions

Scalars, Vectors, and Matrices
Scalars are denoted by lower case letters in italic type. *Vectors* are denoted by bold lower case letters in italic type, as the vector $\boldsymbol{x}$ is composed of elements x_i. *Matrices* are denoted by upper case letters in italic type.

x	scalar
$\boldsymbol{x}$	vector
$\boldsymbol{X}$	matrix
$\boldsymbol{X}^T$	transposed of X
$\boldsymbol{X}^{-1}$	inverse of X
$\boldsymbol{X}^*$	weighted damped Pseudoinverse
$\|\boldsymbol{x}\|$	Euclidean norm
$\overline{\boldsymbol{x}}$	mean value of vector
$f(\cdot)$	scalar function
$\boldsymbol{f}(\cdot)$	vector function
$\dot{x}, \ddot{x}, \ldots$	first, second, ... time derivative of x

Subscripts and Superscripts

$\boldsymbol{x}_0$	initial state at initial time $t_0 = 0$
$\boldsymbol{x}_e$	target state at end time t_e
$\boldsymbol{x}_C$	task descriptor for loop constraint task
$\boldsymbol{x}_{Cj}$	task descriptor for loop constraint j
$\boldsymbol{x}_{Cja}, \boldsymbol{x}_{Cjb}$	task descriptor for loop constraint j from each side of the virtual cut (a and b)
$\boldsymbol{x}_P$	task descriptor for node positioning task
$\boldsymbol{x}_{Pi}$	task descriptor for node positioning task (node i only)
e_{pPi}, e_{oPi}	position and orientation error of node i in node positionining task
$\boldsymbol{x}_S$	task descriptor for shape forming task
$\boldsymbol{x}_A$	task descriptor for active joints task
$\dot{\boldsymbol{x}}_d$	desired value for task descriptor of task k
$\dot{\boldsymbol{x}}_{kd}$	desired task velocity of task k
$\varphi i, max$	maximal joint angle of joint i
$\varphi i, min$	minimal joint angle of joint i

Symbols

General

L	closed loop
l	number of closed loops
M	overall DoF of mechanism
m	number of free nodes of the mechanism (without base node)
N	number of nodes of the mechanism (including base node)
j	number of connections between nodes
f_i	number of DoF of i-th connector
s	step of extension
h_T	node height (Telescopic Element)
w_T	node width (Telescopic Element)
d_T	distance between neighboring nodes (Telescopic Element)
h_C	node height (Cardan Element)
w_C	node width (Cardan Element)
d_C	distance between neighboring nodes (Cardan Element)
$\boldsymbol{p}_S$	surface point on node (to approximate shape)
$\boldsymbol{p}_C$	center point of node
$\boldsymbol{r}$	vector determining normal direction of node
${}^j\boldsymbol{T}_i$	(homogeneous) transformation from frame i to j
$\boldsymbol{I}$	identity matrix
$\boldsymbol{0}$	zero matrix
φ	vector of joint angles
g	inequality constraint function

h	equality constraint function
F	cost function
$\boldsymbol{e}_p$	position error
$\boldsymbol{e}_o$	orientation error
$\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z$	unit vector in x-, y- and z-direction
$\boldsymbol{n}_S$	normal vector of virtual surface
$\boldsymbol{N}$	Nullspace
$\boldsymbol{W}$	weighting matrix
λ	damping factor
$\boldsymbol{\omega}$	angular velocity

List of Figures

List of Tables

1 Introduction

In applications where a human operator is interacting with a remote or virtual environment, all sensory channels of the human operator must be appealed. These senses are sight, hearing, touch, smell and taste. Especially when objects in the environment must be manipulated, sight, hearing and touch are of utmost importance. A human operator interacts with the environment with the help of a human-system interface. While systems for the display of visual and auditory information are already rather developed, providing haptic information by haptic interfaces represents a major challenge in the design of human-system interfaces. One aims for haptic interfaces with a high transparency [5] [6]. A haptic interface is characterized by a high transparency, if the user has the impression of directly interacting with the remote (or virtual) environment and not through a haptic interface. An example for a scenario, where haptic interfaces are used, can be found in the field of telerobotics [7] [8]. Here, a human-system interface is used to control a telerobot intuitively. Visual, auditory, and haptic information is provided by the human-system interface and gives the human operator the impression as if he/she is actually present at the remote environment. The telerobot on the other hand executes the actions of the human operator and conduces as an avatar of the human operator in the remote environment. In other applications, which lie in the field of virtual reality and design, the remote environment is replaced by a simulation. A shape rendering interface (SRI), which is used in the design process of e.g. car bodies, offers the opportunity to display the shape of specific parts, which were previously designed by CAD-software tools. The shape of a part, which is reconstructed by a SRI, can be evaluated and analyzed by exploring it with the bare hands. On the other hand, changes to these parts can be made intuitively by interacting with the SRI. Existing solutions for haptic devices can be divided into tactile displays and kinesthetic interfaces (cf. Figure 1.1).

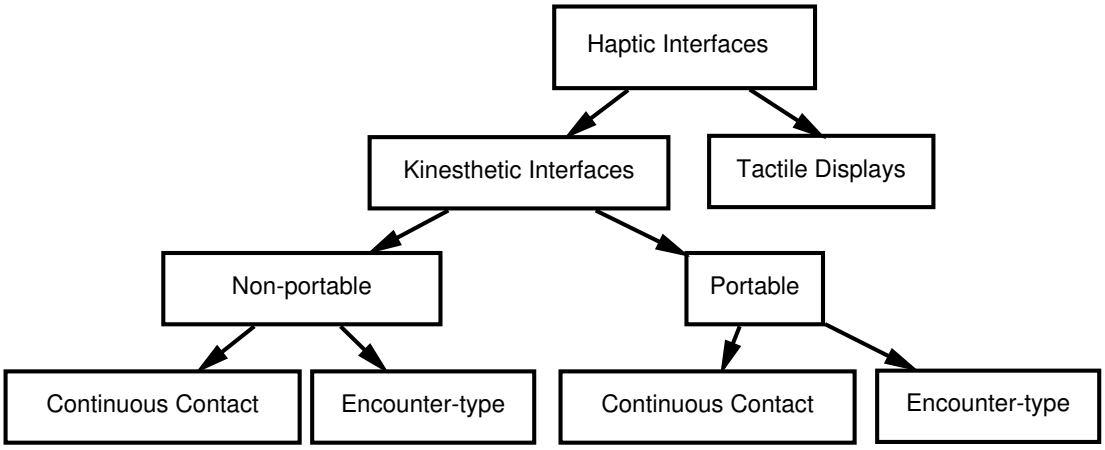


Figure 1.1: Classification of haptic interfaces

Tactile displays are used for delivering information about surface properties. Specific regions on the users palm are therefore stimulated (e.g. with vibrating actuators or by using electric stimulation) in order to give the user the impression as if he/she would touch an object with a specific surface character. A tactile display is, in most cases, not designed for the manipulation of objects. A wide overview of existing tactile displays is presented in [9]. An example of a vibrotactile tactile display and a tactile display which uses electric stimulation to provide tactile feedback is shown in Figure 1.2 and Figure 1.3.

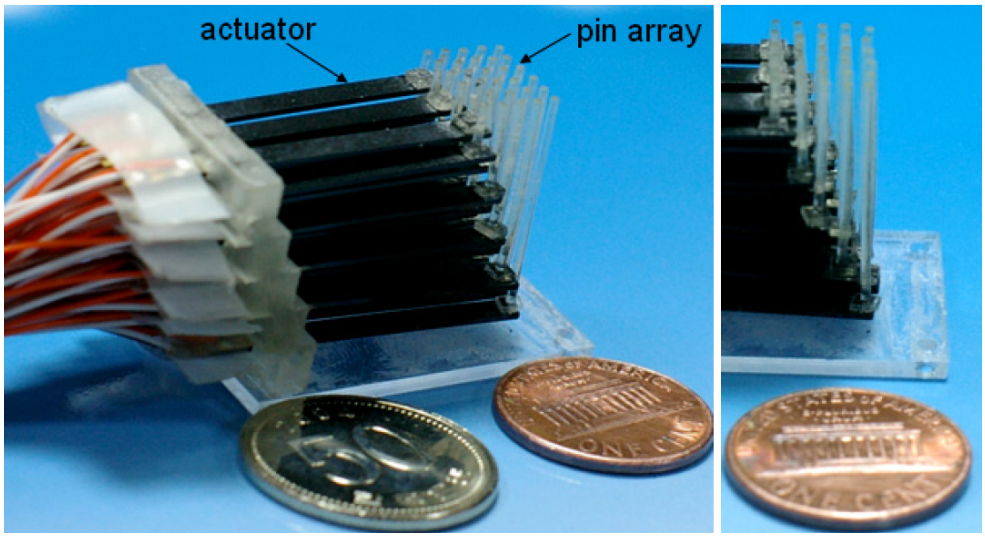


Figure 1.2: Tactile display: vibrotactile pattern [1]. Reprinted, with permission, from K.-U. Kyung, S.-C. Kim, D.-S. Kwon, and M. A. Srinivasan,Texture display mouse kat: Vibrotactile pattern and roughness display, International Conference on Intelligent Robots and Systems, 2006

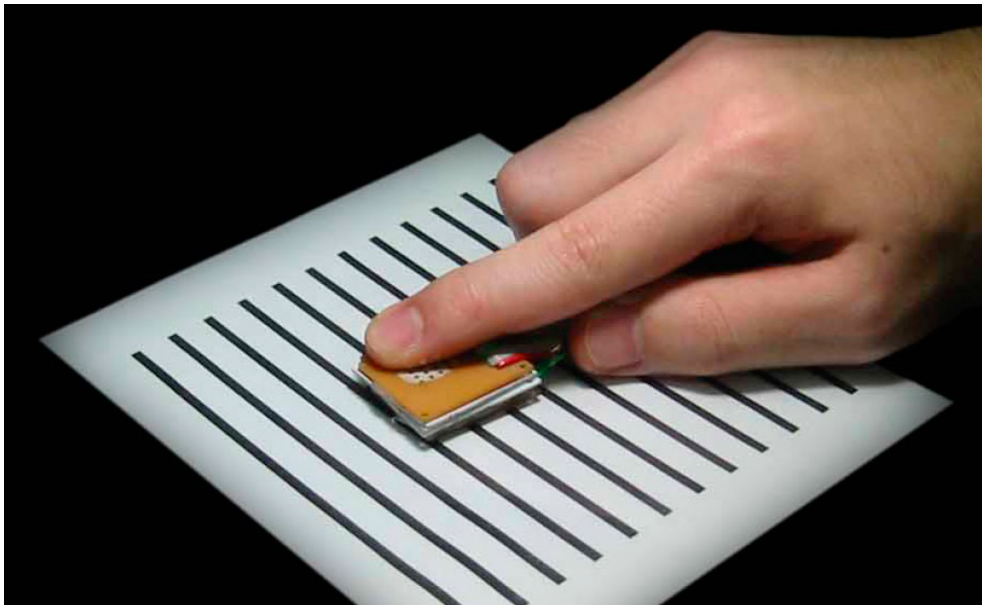
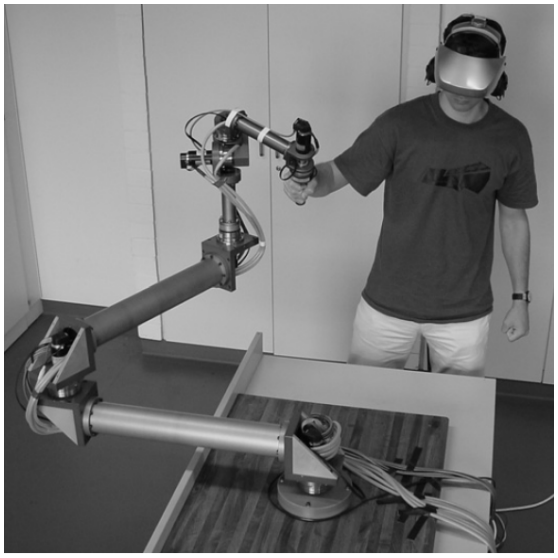
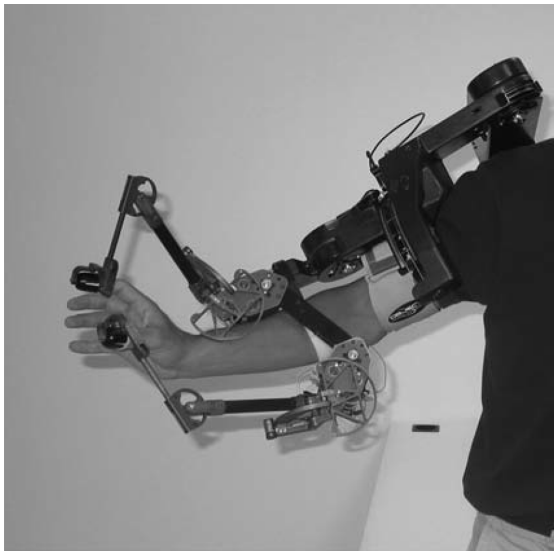


Figure 1.3: Tactile display: electric stimulation [2]. Reprinted, with permission, from H. Kajimoto, N. Kawakami, S. Tachi, and M. Inami, Smarttouch: Electric skin to touch the untouchable, Computer Graphics and Applications, 2004

Kinesthetic interfaces, on the other hand, are used to provide force feedback to the user. Kinesthetic interfaces, see [10] for an overview, can be distinguished in the way they are mounted to the environment. Portable interfaces are mounted to the operator's body in form of an exoskeleton. By doing so, the interface on one hand covers a large workspace, but on the other hand, the transparency is decreased, as the user must carry the weight of the haptic interface. Interfaces, which are mounted to the environment, have a limited workspace, but have the advantage that the weight of the interface is only perceived slightly by the human operator. The workspace of a non-portable interface can be extended by e.g. mounting the interface on a mobile platform [11]. An example of a non-portable and a portable kinesthetic haptic interface is shown in Figure 1.4. A further distinction can be made in the way kinesthetic interfaces are controlled. Here, we differentiate between classical kinesthetic interfaces the user is in continuous contact with and encounter-type interfaces. An encounter-type interface can be regarded as a kinesthetic interface, with the special feature that the user is only in contact with the end-effector of the interface when he really wants to make contact with a remote object (see Figure 1.5) [13] [14] [15] [16]. By doing so, the highest possible transparency is gained for free-space movements. Intention recognition techniques are required to predict the users intention [17] [18] [19] [20]. This is needed because the end-effector has to be positioned by the interface at the corresponding position in space, before the user actually makes contact with it.
Shape rendering interfaces (SRI), also known as 3D shape interfaces, aim for replicating the shape of specific objects or parts of it, in order to provide this information to the user. By combining SRIs with encounter-type interfaces in the sense that a SRI is used as end-effector of the encounter-type display, a very high transparency is gained for the system. The research field dealing with SRIs is still very young. Pioneer work was made in the FEELEX



(a) Non-portable kinesthetic interface: VISHARD 6 [12]. Reprinted, with permission, from M. Ueberle and M. Buss, Design, control, and evaluation of a new 6 dof haptic device, IROS, 2002



(b) Portable kinesthetic interface [10]. Reprinted, with permission, from V. Hayward, O. R. Astley, M. Cruz-Hernandez, D. Grant, and G. Robles-De-La-Torre, Haptic interfaces and devices, Sensor Review, 2004

Figure 1.4: Kinesthetic interfaces

project where one of the first SRIs was developed in 1997 under the name FEELEX 1 [21]. Two examples of SRIs are shown in Figure 1.6.
This thesis shows a systematic analysis for the design and control of 3D SRIs.

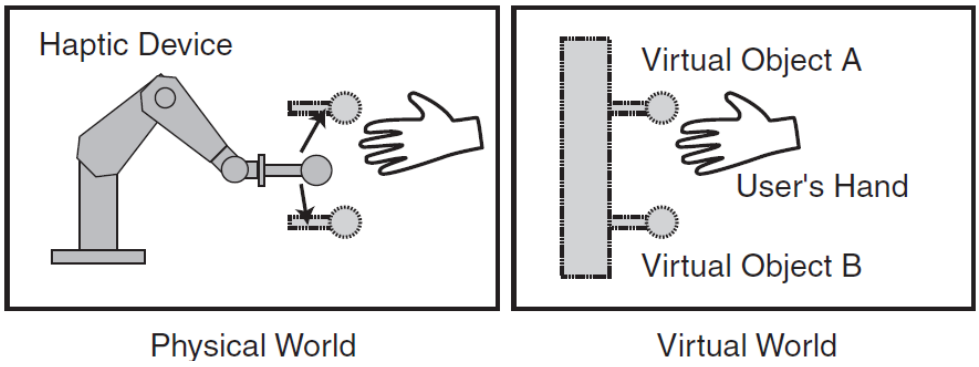
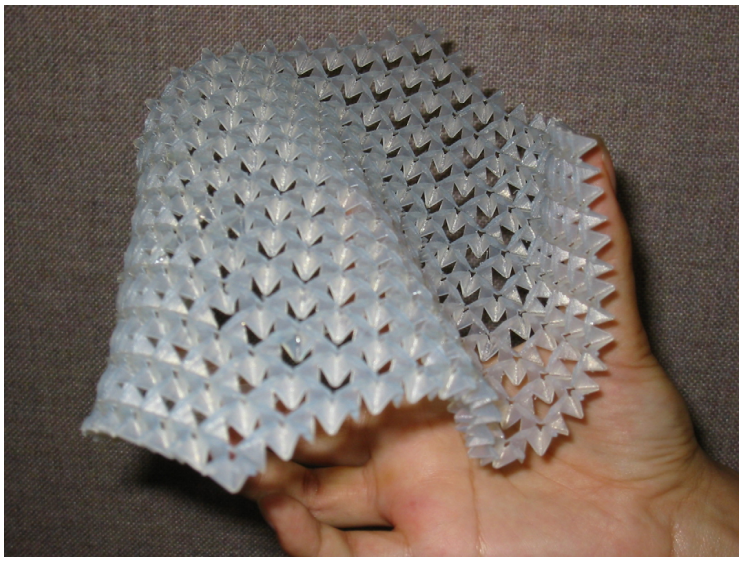


Figure 1.5: Basic idea of an encounter-type interface [3]. Reprinted, with permission, from Y. Yokokohji, J. Kinoshita, and T. Yoshikawa, Path planning for encountered-type haptic devices that render multiple objects in 3d space, Virtual Reality, 2001.

1.1 Problem Definitions and Challenges

When developing SRIs multiple aspects have to be considered in the design process and when controlling the interface. The major challenges are outlined in the following.



(a) Digital Clay: parallel kinematics, unactuated [22]. Reprinted, with permission, from D. Rosen, A. Nguyen, and H. Wang, On the Geometry of Low Degree-of-Freedom Digital Clay Human-Computer Interface Devices, Computers and Information in Engineering Conference, 2003

(b) FEELEX 1: pin array, actuated [21]. Reprinted, with permission, from H. Iwata, H. Yano, F. Nakaizumi, and R. Kawamura, Project FEELEX: adding haptic surface to graphics, Computer graphics and interactive techniques, 2001

Figure 1.6: Shape rendering interfaces

In [23] Hayward and Astley define performance measures for kinesthetic haptic interfaces, which are limitedly applicable to SRIs: Degrees of Freedom, Device-Body Interface, Motion Range, Peak Force, Inertia and Damping, Peak Acceleration, Energy Flux - Power Density, Resolution, Precision, Bandwidth, Structural Response, Dynamic Precision and Closed Loop Performance.
Only few performance measures are defined for SRIs and some of the measures enumerated above are not suitable for them. For SRIs especially *Degrees of Freedom*, *Motion Range*, *Peak Force* and *Resolution* represent the most important performance measures and thus, extra attention must be paid to them. Most of the existing kinesthetic interfaces feature a single-point interaction. In this case, the control of one point, the point of the end-effector, is of interest. In contrast to that, SRIs deal with the control of a whole surface, the surface, which renders a virtual object. While for most kinesthetic interfaces 6 degrees of freedom (DoF) are sufficient, because the position and orientation of one point can be completely described with it, SRIs need more DoF in order to describe the profile of the surface, which is rendered. In other words, multiple points on the surface must be described by multiple solid bodies within the mechanism. Because of this, far more than 6 DoF are needed to render a shape sufficiently precise, which consequently leads to a highly redundant mechanical design.
The motion range of a SRI highly contributes to the ability of the interface to render a wide variety of shapes. It is clear that this performance measure is dependent on other performance measures like the DoF and the detailed design of the mechanism.

As haptic interfaces (SRIs included) must display soft as well as stiff objects, a high peak force is requested for them.
Whether a SRI is able to display filigree shapes or not is strongly dependent on its spatial resolution. We define the spatial resolution by the distance of two neighboring surface points, which can be rendered by the interface. The spatial resolution is not to be confused with the resolution defined in [23]. In here, "the resolution is defined as the smallest deviation from system equilibrium which can be detected by the sensors under study".
The foregoing thoughts lead to a design where a high amount of DoF is concentrated on very limited space. A high amount of joints must be actuated by strong actuators and sensed by precise sensors. These hardware components must be built in at very limited space. In summary, one can say that the major challenge in the design of SRIs is to pack a high amount of DoF and strong actuators in very limited space without restricting the motion of range of the mechanism.
The control of SRIs is another major challenge when dealing with them. A framework to handle the aforementioned redundancy in real-time must be developed. As mentioned before SRIs feature a high amount of DoF, which are actuated by a large number of actuators. The DoF must be actuated conjointly in order to obtain a requested shape. Especially when dealing with a parallel kinematics, which features high stiffness and speed, the dependency of passive (unactuated) joints on active (actuated) joints represents a major challenge when controlling the device. The forward and inverse kinematic description is in most cases not available, and only time-consuming numerical solutions do exist. Even if a kinematic solution is available, the optimal positioning of the mechanism to render a shape is an open question. Multiple configurations of the mechanism might represent a solution to the shape-rendering problem, as in most cases we deal with a redundant kinematics.
A further challenge within the field of SRIs is the scalability and extendibility of the mechanism. As said before, a SRI must feature a high amount of DoF/actuators/sensors on limited space. When scaling down a SRI to gain a higher spatial resolution the space per DoF/actuator/sensor is reduced further. On the other hand, one wants an interaction area, which has an appropriate size. When scaling down an existing mechanism one wants to keep or extract the size of the interaction area. Consequently, the mechanism must be extended and the amount of actuators increased. Finding appropriate proceedings to reduce the amount of needed actuators represents a further challenge when dealing with SRIs.

1.2 Main Contributions and Outline of the Dissertation

The main goal of this thesis is to present a systematic analysis for the design and control of SRIs. The design must have a high resolution, must be easily extendable and, in contrast to most state-of-the-art interfaces, must be actuated.
The thesis is organized in five chapters. Chapter 2 presents the design of a SRI. Chapter 3 deals with the kinematics of SRIs and Chapter 4 gives directions to reduce the amount of needed actuators. The thesis concludes with Chapter 5, which summarizes the most important results and formulates directions of future work.
In Chapter 2 the design of a SRI, the Formable Object (FO), is presented. The interface has 24 DoF, is designed as a parallel kinematics and can easily be extended. The FO has, in op-

position to comparable state-of-the-art interfaces, a higher resolution and is fully actuated. After defining requirements for the interface, basic elements, from which the mechanism is assembled, are presented and their use for the mechanism is motivated. The smallest step of extension for the mechanism is defined and the determinacy is proofed for all steps of extensions. A numerical procedure for the quantitative determination of the formability of the mechanism is presented. It is used to find the optimal configuration of combined basic elements in terms of formability. A preselection of configurations is done a priori to reduce the needed time for finding the optimal configuration of the SRI. After this configuration is determined, a detailed description of the design including actuation and sensing principles is presented and a realized prototype is introduced. The prototype is evaluated in terms of maximum and minimum displayable curvatures, stiffness and dynamic performance.
Chapter 3 deals with the kinematics of SRIs especially with interfaces, which feature a parallel kinematic design. As mentioned earlier, no real-time kinematic solutions are known for SRIs. After defining the problem of shape rendering, aspects, which have to be considered in the problem solution, are identified. The kinematics problem is solved in a differential way. The procedure of Nullspace projection outlined in [24], [25] or [26] is modified to the problem of shape rendering. Therefore, specific tasks namely the *Loop Constraint* task, the *Node Positioning* task and the *Shape Forming* task are introduced, which are executed on the Nullspace of the corresponding higher order task. Two modes of shape rendering are distinguished: with or without user interaction. Furthermore, different shape descriptions are discussed, namely implicit surfaces and polygon sets. It is shown that the introduced framework can be used similarly for both shape descriptions. The chapter concludes with simulations of the different control modes and shows their differences.
In Chapter 4 actions to reduce the needed amount of actuators are discussed. Using one actuator, which sequentially can be connected to the actuated joints of the mechanism with the help of clutches, represents a practical procedure to reduce the amount of actuators and therewith to minimize hardware costs. By doing so, the sequence in which the actuators are controlled is not trivial, as the passive joints are dependent on the active joints. Not taking care of the switching sequence and switching times could drive the passive joints into their joint limits and the demanded shape cannot be reached or, in the worst case the mechanism breaks. Thus, the system is defined as a hybrid system and the finding of the switching sequence is formulated as a hybrid control optimization problem. After performing simplifications, the problem can be formulated as purely discrete problem and is solved with combinatorial graph search approaches. The approach is demonstrated on a planar parallel kinematics with 6 DoF. The found solutions are discussed in terms of their optimality.
Finally Chapter 5 summarizes the main results of this thesis and elaborates directions of future research.

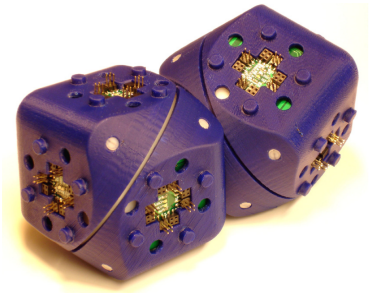
2 Design of a Shape Rendering Interface: The Formable Object

Interacting with 3D-shape objects using bare hands represents a very intuitive way to explore object shapes and offers many new opportunities in the field of virtual reality and design. Only a few 3D-shape interfaces are known in literature so far, but their resolution is rather low and/or they are not actuated. Thus, a new parallel kinematic design for an actuated 3D-shape interface with a comparatively higher resolution is presented, which is fully actuated and which can be further extended to cover larger interaction areas. Starting from a preliminary configuration, the kinematics is optimized for the rendering of basic shapes like cylinders or spheres. Due to its specific parallel kinematic design, where a determined system is gained by attaching one node to the environment, it can easily be mounted as an end-effector to kinesthetic haptic interfaces. The prototype of the Formable Object (FO) is evaluated concerning its ability to render shapes, along with stiffness measurements and its capability for dynamic shape rendering.
Haptic interfaces for shape rendering are either devices that can morph their shape to form a surface to be explored by a user, or devices that generate the illusion of experiencing a real shape. Haptic interfaces for shape rendering promise applications in the field of design and prototyping. By using a shape rendering interface (SRI) in the design process of cars or household aids for example, shapes can be visualized and haptically explored without costs for manufacturing real parts. The shape/design of a product body highly contributes to the commercial success of it. With a SRI, shapes can be evaluated in a very intuitive way and modifications can be made with bare hands and directly applied to the design. Further, by using a SRI, the surface of objects can be explored with bare hands in contrast to state-of-the-art interfaces where only point-contacts can be simulated. A SRI is especially useful in combination with an encounter-type haptic interface [13], [14], [15], [16], [27], [28], [3], [29], [30]. Encounter-type haptic interfaces are interfaces where a user is only in contact with when contact should be rendered. The end-effector of the encounter-type interface, which represents an object in the user's environment, is positioned at the corresponding position in the environment waiting for the user to make contact. Assuming that the user's intention can be predicted to a certain level, the encounter-type interface can be used to represent multiple objects in the environment. Using a SRI as end-effector will make the encounter-type interface universally applicable, as the different shapes of the objects can be felt by the user. However, only few mechanical concepts have been developed up to now. Basically three design concepts can be distinguished: a) Formable Body Concepts, b) Formable Crust Concepts and c) Non-contact Displays. These concepts will be described in the following paragraphs.

Formable Body Concepts: The Claytronics project [31] deals with a very preliminary formable body concept. The main idea is, that 3D-shapes are generated by using pro-

grammable manner (modular robots). The manner consists of catoms (claytronic atoms), which can move in three dimensions in relation to others or adhere to other catoms to form a 3D shape. With the claytronic concept very complex shapes are possible (e.g. holes in the object). The main drawback, however, is that at present the catoms are bulky (45mm in diameter), thus the resolution is unsatisfying. The prototype shown in Figure 2.1a can only display shapes in 2D.
A low cost version of a similar approach is presented in [32]. The so-called molecubes shown in Figure 2.1b have a diameter of 66mm and can form shapes in 3D.

(a) Claytronic Atoms [31]. Reprinted, with permission, from S. Goldstein, J. Campbell, and T. Mowry, Programmable matter, Computer,2005.



(b) Molecubes [32]. Reprinted, with permission, from V. Zykov, A. Chan, and H. Lipson, Molecubes: An open-source modular robotics kit, IROS, 2007

Figure 2.1: Two formable body concepts

Formable Crust Concepts: The FEELEX mechanism [21] can be regarded as a 2.5D-formable crust concept (Figure 2.2a). The FEELEX uses a linear actuator array. Each actuator drives a rod, which is located underneath a rubber membrane, which is deformed by the rods. A drawback of this concept is that it is equipped with a bulky actuator arrangement and it is not possible to display shapes with undercuts. A similar approach was already outlined in [33].

In [34] multiple formable crust concepts for deforming a membrane or strip by bending it with actuators located directly underneath the membrane/strip are discussed. However, these concepts are only theoretical.

A preliminary concept is Digital Clay developed at Georgia Institute of Technology [22](Figure 2.2c). The principle item of Digital Clay is a multiple collocated spherical joint developed by Bosscher et al. [35], which can easily be manufactured in a very small scale. By combining the spherical joints to an array, a formable crust is obtained. However, the actuation of the Digital Clay is an unsolved problem.

Bordegoni et al. [36] developed a Self-Deformable Haptic Strip, which uses bending and torsion modules to deform a plastic strip (Figure 2.2b). A prototype of the mechanism exists. A drawback of the Haptic Strip is the missing extendibility of the mechanism. The Haptic Strip can display a narrow band of the surface only and exploration of larger objects is only

possible if the Haptic Strip is moved or rotated.

Mazzone [37] [38] developed a parallel kinematics, the so-called Smart Mesh, where multiple nodes are connected to a network of end-effectors to display universal shapes (Figure 2.2d). The mechanism is not actuated and has a rather low resolution (average node distance: 195mm).

In [39] we presented a preliminary version of our Formable Object, which is described in this chapter.

Non-contact Displays: Non-contact displays are often classified as tactile displays. Like SRIs, they create haptic stimuli in 3D-space at multiple points of the users hand and are therefore comparable to them to a certain extent. Such displays use air-flow or ultrasonic transducers to provide haptic sensations. Hoshi et al. [4] developed an array of ultrasonic transducers, which are able to provide haptic feedback in space above the array, but grasping or manipulation of virtual objects is not possible (Figure 2.3).

As this thesis outlines the design of a formable crust concept, in Table 2.1 a comparison of existing analogues [21], [22], [36] and [38] is shown.

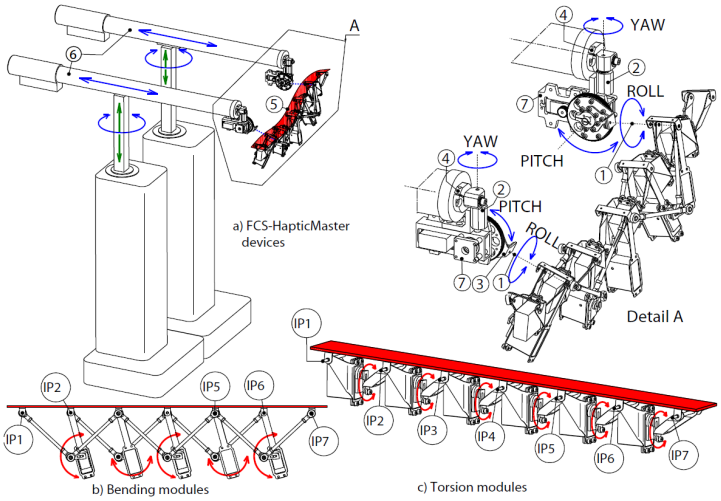
Tabular 2.1: Comparison of existing analogues

	FEELEX [21]	Digital Clay [22]	Haptic Strip [36]	Smart Mesh [38]
Average node distance	4.8cm	1.3cm	10cm (estimated)	19.5cm
Number of nodes	36	> 100	6 (only one direction)	16
Undercuts	no	yes	yes	yes
Actuated	yes	no	yes	no

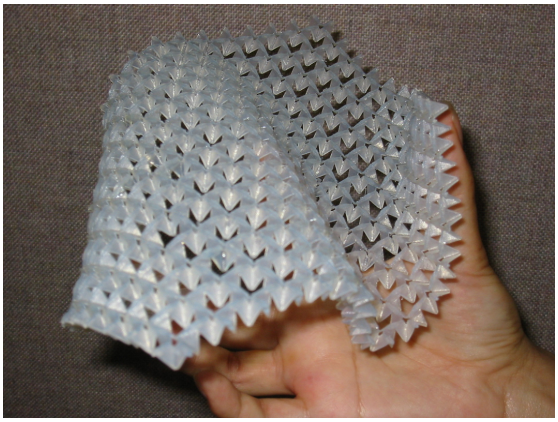
In this chapter a design for a formable crust, in the following called Formable Object (FO), is presented. The FO is not only a theoretical concept, but, compared to several concepts presented above, is designed to build a prototype with a comparatively higher resolution than state-of-the-art developments. The FO can be mounted at the end-effector of a robotic arm, which is important when the FO is to be moved to different positions in space. The FO has 24 degrees of freedom (DoF) which allows displaying basic shapes like cylinders or spheres.

In Section 2.1 preliminary considerations are made which lead to the optimal configuration of the FO. In Section 2.2 details about the mechanical construction including actuation and sensing principles and the control of the FO will be presented. In Section 2.4 the FO is evaluated. Maximum curvatures, stiffness and the required time to render test shapes are determined and discussed. Finally, this chapter concludes with a summary and outlook to future work.

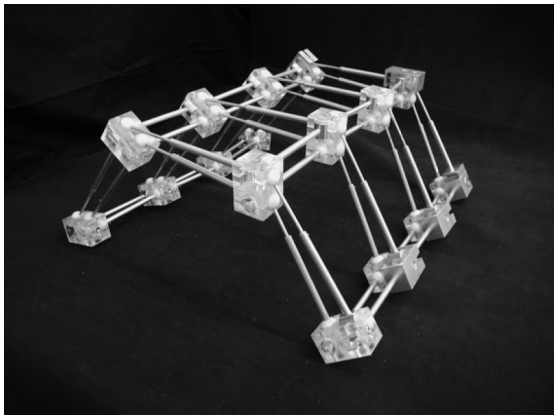
(a) FEELEX [21]. Reprinted, with permission, from H. Iwata, H. Yano, F. Nakaizumi, and R. Kawamura, Project FEELEX: adding haptic surface to graphics, Computer graphics and interactive techniques, 2001



(b) Haptic Strip [36]. Reprinted, with permission, from [M. Bordegoni, U. Cugini, M. Covarrubias, and M. Antolini, A Force and Touch Sensitive Self-deformable Haptic Strip for Exploration and Deformation of Digital Surfaces, Haptics: Generating and Perceiving Tangible Sensations, 2010



(c) Digital Clay [22]. Reprinted, with permission, from D. Rosen, A. Nguyen, and H. Wang, On the Geometry of Low Degree-of-Freedom Digital Clay Human-Computer Interface Devices, Computers and Information in Engineering Conference, 2003



(d) Smart Mesh [38]. Reprinted, with permission, from A. Mazzone, C. Spagno, and A. Kunz, A haptic feedback device based on an active mesh, ACM symposium on Virtual reality software and technology, 2003

Figure 2.2: Formable crust concepts

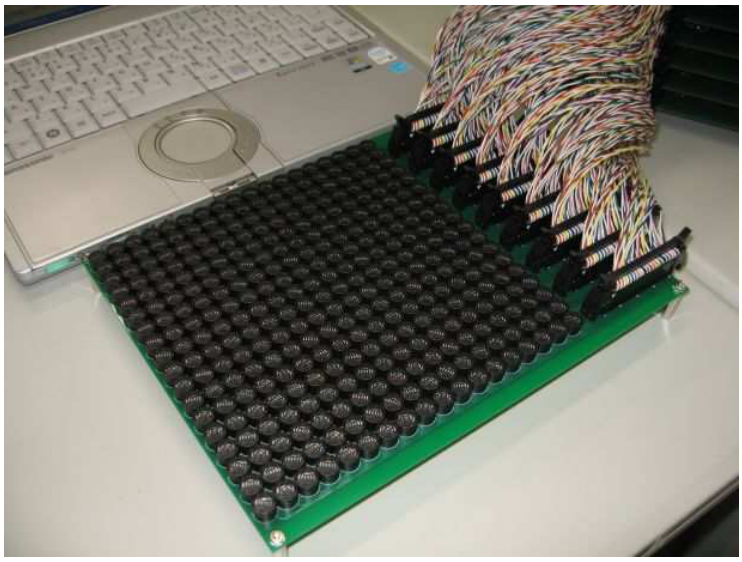


Figure 2.3: Non-contact display [4]. Reprinted, with permission, from T. Hoshi, T. Iwamoto, and H. Shinoda, Non-contact tactile sensation synthesized by ultrasound transducers, EuroHaptics conference, 2009

2.1 Mechanical Design Considerations

2.1.1 Requirements

In this section, requirements for the FO are stated and preliminary design measures are derived. As the FO is to be used as end-effector of an encountered-type haptic interface (cf. [3]), it must be lightweight and easily mountable. In order to be able to render stiff objects, high stiffness of the mechanical structure is demanded ($\approx$ 1 kN/m). The maximum force of a pinch grasp is about 60 N [40]. Working against such high forces, however, is not required, because most object manipulations are not performed with maximum forces, so we decided to set the maximum demanded counterforce in normal direction of the FO-surface to 10 N. The resolution should be as high as possible, guaranteeing also actuation of the whole device. Simple shapes like cylinders or spheres (concave & convex) must be presentable by the interface. Furthermore, the expandability to interfaces, which cover larger surfaces (more DoF), should be guaranteed. In this work, we deal with kinematics where a shape is represented by nodes, which are solid bodies within the mechanism. However, we want to render continuous surfaces. Thus, the nodes/kinematics must be covered by an elastic layer, which bridges the gaps between the nodes. The layer must be stiff enough so that the regions between the nodes are not perceived too soft and elastic enough so that the mechanism is not limited in its motion. For a first version we propose to use a highly elastic layer, which can be achieved by a thin layer of silicon or a thermoplastic elastomer. In future versions we aim for a layer with controllable elasticity.

2.1.2 Determinacy

For simplicity, we aim for a square grid crust, which means that a mechanism consisting of nodes and connections between these nodes is arranged in a square grid. Figure 2.4 shows a simple square grid with 9 nodes where node 1 is the base node fixed to the environment. It contains 4 closed kinematic chains $(L_1, \ldots, L_4)$. The connections between the nodes, which are used to render the shape, feature several joints. In [41] we showed that 3 degrees of

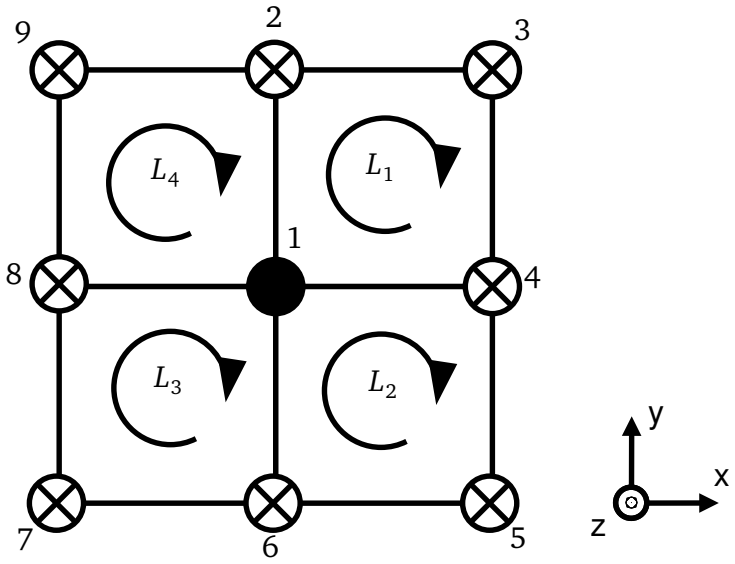


Figure 2.4: Square grid with 9 nodes

freedom (DoF) per node are needed to render a shape. This means that $M = 3m$ must hold, with m the number of free nodes and M the number of overall DoF of the device. A similar statement is given in [34] where the *Surface Freedom Measure* (SFM) is introduced. The SFM relates the number of overall DoF to the number of surface points. The highest possible value of the SFM is 3.
The overall DoF M of a mechanism can be calculated using the well-known Chebychev-Grübler-Kutzbach criterion [42]

$$M(s) = 6m(s) - \sum_{i=1}^{j}(6 - f_i) = 6(N(s) - 1 - j(s)) + \sum_{i=1}^{j} f_i \tag{2.1}$$

where M determines the overall degrees of freedom (DoF), m is the number of rigid bodies, j the number of connections and f_i the number of joints of the i-th connection. N is the number of rigid bodies m plus the fixed base ($N = m + 1$). The step of extension s is important when analyzing the expandability of the FO, what will be done in Section 2.1.5. For the square grid crust shown in Figure 2.4, $m = 8$ and $j = 12$. The overall DoF must be $M = 3m = 24$ in order to be able to render shapes. Inserting these values into (2.1) one receives $\sum_{i=1}^{j} f_i = 48$, which means that 48 joints are needed for the mechanism. We decided to equally distribute these joints over the 12 connections which brings us to $f_i = 4$ joints per connection. In order to gain a statically determined mechanism, the number of overall DoF M must be equal to the number of actuators a. This means that 24 actuators are needed. We also decided to

distribute the actuators equally over the 12 connections, to partition the available space for the actuators equally. This means that 2 joints per connection are actuated.

2.1.3 Basic Elements

The FO is assembled of multiple basic elements with 4 DoF each. We distinguish two types of elements: The *Telescopic Element* and the *Cardan Element*. The structures presented in the following sections are assembled from these basic elements.

Telescopic Element

A schematic of the Telescopic Element is shown in Figure 2.5. Two nodes are connected by a telescopic rod with 2 axial bearings on its ends. The axial bearings deliver 2 degrees of freedom (DoF) and the telescopic rod, which can extend and rotate about its own axis, delivers another 2 DoF. Thus, the connection between two nodes has 4 DoF.

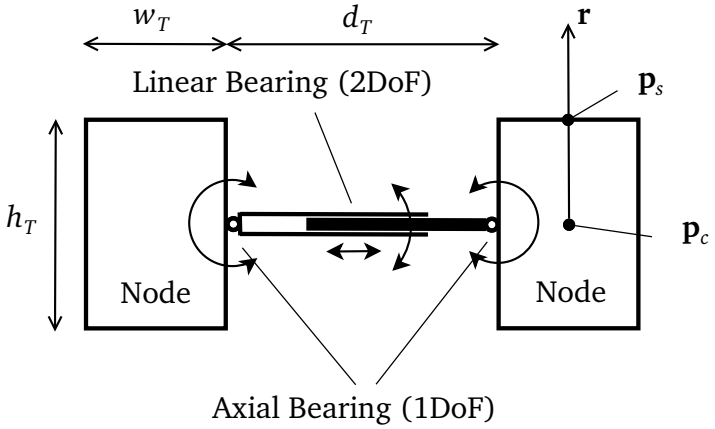


Figure 2.5: Side view of Telescopic Element with 4 DoF. (h_T: node height, w_T: node width, d_T: distance between nodes, $\mathbf{p}_c$: node center, $\mathbf{p}_s$: node surface point, $\mathbf{r}$: direction vector)

Cardan Element

The Cardan Element is sketched in Figure 2.6. It has an asymmetric design and does not include a telescopic rod. A small rod, which is able to rotate relative to the node, which it is mounted to, is connected to a Cardan joint. This delivers 3 DoF similar to a ball joint. The Cardan joint is further connected to a rod, which connects to the connecting node via an axial bearing, which delivers another degree of freedom. Thus, the Cardan Element has 4 DoF, equal to the Telescopic Element.

2.1.4 Kinematic Design of Basic Configuration

In this section we aim to find an optimal distribution of basic elements within the design with 9 nodes (cf. Figure 2.4). We therefore compare different configurations with regard to

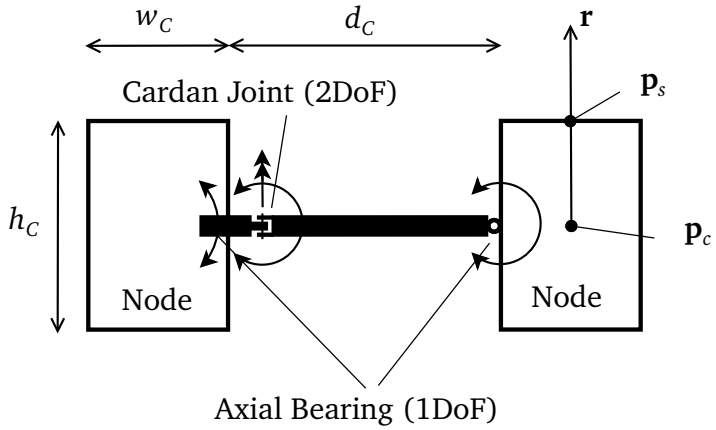


Figure 2.6: Side view of Cardan Element with 4 DoF. (h_C: node height, w_C: node width, d_C: distance between nodes, $\mathbf{p}_c$: node center, $\mathbf{p}_s$: node surface point, $\mathbf{r}$: direction vector)

their ability to render some basic shapes, in the following referred to as *formability*. In [39] we simulated a configuration of the FO, which solely contained Telescopic Elements. We observed that especially spherical shapes were not rendered satisfactorily. The nodes were not able to position and orient properly at the same time. As shown in [41] 3 DoF per node are needed to render a shape. To be more precise we need one translational DoF (distance to the virtual surface) and two rotational DoF (alignment normal to virtual surface) per node. Thus, some translational DoF must be replaced by rotational DoF. On the other hand, in [34] authors propose to introduce prismatic joints to increase the formability of a grid crust. Thus, in order to allow for stretching of the mechanism, not all Telescopic Elements should be replaced by Cardan Elements. We have 12 connectors distributed between the 9 nodes of a basic structure where Telescopic or Cardan Elements can be installed. As the Cardan Element can be installed in two different directions we have 3 possible configurations per connection and $3^{12} = 531441$ configurations for the whole device. For the comparison of different configurations the *Formability Measure* (FM) is introduced in this section. With the FM a quantitative statement about the formability of a configuration can be made. In terms of expandability, we favor a symmetric design configuration to get similar characteristics in all directions of the mechanisms surface. With a symmetric design, the gained knowledge at a certain design step can be used for any step of extension. Further, as the determination of the FM for one configuration takes several hours, it is not practical to compare every single configuration. Thus, we decided to reduce the amount of configurations, by considering only symmetric configurations. This reduces the amount of possible configurations to 9 (see Figure 2.7).

In order to find the optimal solution out of these nine configurations we compare them in terms of their formability. In [34] some related performance measures for a digital clay are defined: the *Surface Freedom Measure* (SFM), the *Freedom from Ground Measure* (FGM) and the *Data Resolution* (DR). The SFM relates the number of DoF to the surface points and the highest possible value is 3. The FGM is a measure of the average distance that a surface point has from the ground in terms of joint freedoms. There is no upper bound for the FGM and higher values are preferred, but if the FGM is too high the structure may have problems

to maintain rigidity. The *DR* relates the number of DoF to the used actuators and my range from 0 to 1 with 1 the best possible value. For a detailed description on these performance measures, the reader should refer to [34]. All these measures are based on the amount of DoF and nodes and do not take into account the precise configuration of a mechanism. Thus, all configurations in Figure 2.7 have the same values for SFM, FGM and DR. For integrity reasons we want to give the values for these performance measures for the FO: $SFM = 3$, $FGM = 6$ and $DR = 1$.

Before comparing the different versions quantitatively, a qualitative analysis should be given.

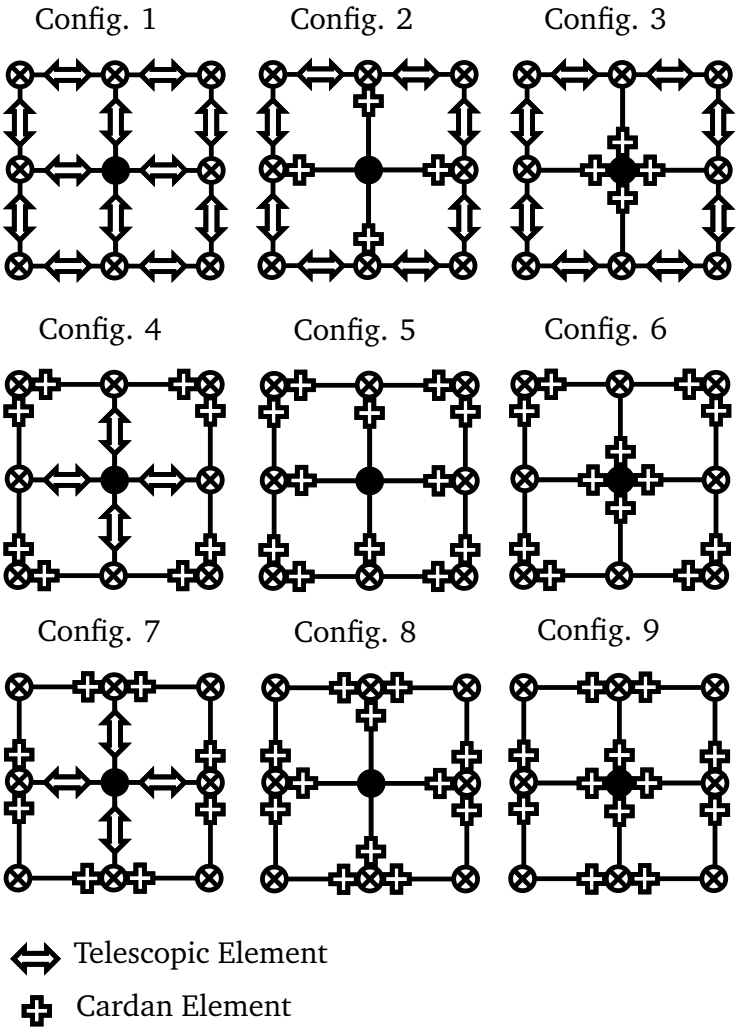


Figure 2.7: Different assemblies of the square grid design

As already mentioned, a combination of Telescopic and Cardan Elements and not only one of the two types must be installed. When reforming from a flat configuration to a sphere some connections have to change length (Telescopic Elements) in order to form the curved surface of the sphere. Telescopic Elements are not implicitly needed to render cylindrical shapes because only one bending direction exists in this case. Configuration 1 has only Telescopic Elements built in. In configuration 2 and 3 the Cardan Elements are distributed over the connectors, which connect the base node (node 1) with the remaining nodes. Configurations 4 and 7 have Cardan Elements in the surrounding connectors and Telescopic Elements connect

to the base node. Configuration 5, 6, 8 and 9 use Cardan Elements only. We expect bad shape rendering results for configuration 1, 5, 6, 8 and 9 because only one of the two basic element types is used. Configurations 4 and 7 are preferred over configurations 2 and 3 because of the following reason: When rendering shapes the loop constraints ($L_1 \dots L_4$, cf. Figure 2.4) must hold. In all configurations, nodes 2, 4, 6 and 8 have 4 DoF relative to the base node (when considered without nodes 3, 5, 7 and 9). As mentioned before 3 DoF per node are needed to render a shape. This means that 1 DoF is redundant and can be used to find a suitable configuration of the FO so that nodes 3, 5, 7 and 9 also render the shape properly. In configuration 2 and 3 this remaining DoF is a rotational DoF in z-direction which does not change the position of the node (2, 4, 6, 8), but rather distorts the adjoining closed loops. Thus, this DoF is important for nodes 3, 5, 7 and 9, but will in most cases remain unused in order to not violate the closed loop constraints. This is not the case with configuration 4 or 7, because the redundant DoF for node 2, 4, 6 and 8 can be regarded as a translational DoF. This DoF can be used to position nodes 2, 4, 6 and 8 so that nodes 3, 5, 7 and 9 can better render the shape. In the following we quantitatively compare the 9 configurations.
In [39] we presented a numerical method for the inverse kinematics of the FO, which we want to recapitulate here: The shape-rendering problem is formulated as a constrained optimization problem. The decision space of the optimization is defined by the four closed loops $(L_1, \dots, L_4)$, which specify the nonlinear equality constraints

$$L_1 : \underbrace{{}^1\boldsymbol{T}_2 \cdot {}^2\boldsymbol{T}_3 \cdot {}^3\boldsymbol{T}_4 \cdot {}^4\boldsymbol{T}_1}_{\boldsymbol{T}_1} - \boldsymbol{I} = \boldsymbol{0}, \tag{2.2}$$

$$L_2 : \underbrace{{}^1\boldsymbol{T}_6 \cdot {}^6\boldsymbol{T}_5 \cdot {}^5\boldsymbol{T}_4 \cdot {}^4\boldsymbol{T}_1}_{\boldsymbol{T}_2} - \boldsymbol{I} = \boldsymbol{0}, \tag{2.3}$$

$$L_3 : \underbrace{{}^1\boldsymbol{T}_6 \cdot {}^6\boldsymbol{T}_7 \cdot {}^7\boldsymbol{T}_8 \cdot {}^8\boldsymbol{T}_1}_{\boldsymbol{T}_3} - \boldsymbol{I} = \boldsymbol{0}, \tag{2.4}$$

$$L_4 : \underbrace{{}^1\boldsymbol{T}_2 \cdot {}^2\boldsymbol{T}_9 \cdot {}^9\boldsymbol{T}_8 \cdot {}^8\boldsymbol{T}_1}_{\boldsymbol{T}_4} - \boldsymbol{I} = \boldsymbol{0}. \tag{2.5}$$

In (2.2) - (2.5) ${}^j\boldsymbol{T}_i = {}^j\boldsymbol{T}_i(\boldsymbol{\varphi})$ specifies a transformation from frame i to frame j, depending on all joint angles $\boldsymbol{\varphi}$. These frames are body fixed to the corresponding node. We use homogeneous transformations, but other descriptions like e.g. Quaternions can be used as well. As each loop ((2.2) - (2.5)) provides 12 equations (9 rotation, 3 position), 48 equality constraints $h_l(\boldsymbol{\varphi})$ must hold. The joint limits of the linking elements specify the inequality constraints. As each joint has one maximum and one minimum value and 48 joints are built in, 96 inequality constraints $g_k(\boldsymbol{\varphi})$ must hold. Note that this mathematical description can easily be modified for different kinematics with more or less loops and joints, as only the amount (and form) of equality and inequality constraints must be adapted. The optimization problem is formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{\varphi}} \quad & F(\boldsymbol{\varphi}) = k_o \frac{F_o(\boldsymbol{\varphi})}{\pi/2} + k_p \frac{F_p(\boldsymbol{\varphi})}{1\mathrm{m}} \\ \text{subject to} \quad & g_k(\boldsymbol{\varphi}) \le b_k,\ k = 1, \dots, 96, \\ & h_l(\boldsymbol{\varphi}) = 0,\ l = 1, \dots, 48. \end{aligned} \tag{2.6}$$

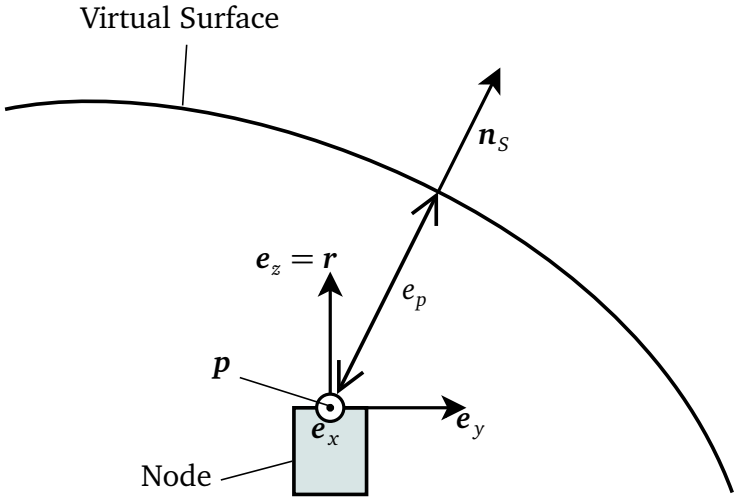


Figure 2.8: Position and orientation error of node

In (2.6) F_p and F_o is the Euclidean norm of the position and orientation errors ($\boldsymbol{e}_p = \left[e_{p2} \dots e_{p9}\right]^T$ and $\boldsymbol{e}_o = \left[e_{o2} \dots e_{o9}\right]^T$) of the 8 free nodes. On each node i a surface point $\boldsymbol{p}_i$ is defined, which is used to render the shape of interest. The orientation of the node surface at $\boldsymbol{p}_i$ is defined as $\boldsymbol{r}_i$. The position error e_{pi} of node i is defined as the Cartesian distance of $\boldsymbol{p}_i$ to the virtual surface which is to be rendered (cf. Figure 2.8). The orientation error e_{oi} of node i is defined as the angle between $\boldsymbol{r}_i$ and the normal vector of the surface $\boldsymbol{n}_S$. The weighting factors k_p and k_o are used to weight position and orientation errors. The errors are normalized to 1 m and $\pi/2$, respectively. For the optimization the *interior-point*-algorithm [43] is used. All optimizations were executed in Matlab® 2012b. The optimization was performed on an Intel Core i5-2500K Quad-Core-Processor (3.3GHz/7.5GB). A maximum number of 2.000 iterations was selected. A flat configuration of the FO was used as starting configuration for the optimization. Note that the kinematics (configuration 1 to 9) is reflected in the equality constraints. The optimization cannot run in real time, but changes in the kinematics can be made easily and thus, multiple configurations can be tested in short time.

In order to compare diverse configurations quantitatively, we introduce the *Formability Measure* FM:

$$FM = \int_{p_1} \int_{p_2} \dots \int_{p_n} F(\varphi)\, dp_n \dots dp_2\, dp_1 \tag{2.7}$$

In (2.7) the integral of the normalized node errors as defined in (2.6) is built over the parameter space with n parameters, which define the shapes of interest. The Formability Measure FM can take values between zero and infinity, while zero denotes perfectly rendered shapes (no position and no orientation errors) over the whole parameter space. The unit of FM depends on the parameter space which is analyzed. In the following, we want to compare the 9 configurations mentioned earlier with regard to elliptic shapes. The implicit form of a 3D-ellipsoid is:

$$f(x,y,z) = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 \tag{2.8}$$

with a, b and c the semi-axes of the ellipsoid. Note that spheres ($a = b = c$) and cylinders (e.g. $a = \text{inf}$, $b = c$) are special cases of the ellipsoid. The 9 configurations are analyzed over the parameter space of $a = 0.08m \ldots 0.5m$ and $b = c = 0.08m \ldots 0.5m$. The center node (node 1) is positioned at position $\boldsymbol{p}_1 = \begin{bmatrix} 0 & 0 & c \end{bmatrix}^T$ with orientation $\boldsymbol{r}_1 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. The Formability Measure with parameters $p_1 = a$ and $p_2 = b = c$ has the following form:

$$FM = \int_{p_{1,min}}^{p_{1,max}} \int_{p_{2,min}}^{p_{2,max}} F(\varphi)\, dp_2\, dp_1 \tag{2.9}$$

Tabular 2.2: Comparison of configurations

Config.	1	2	3	4	5	6	7	8	9
$FM\left[\frac{1}{m^2}\right]$	0.0684	0.0729	0.0723	0.0048	0.0369	0.0372	0.0053	0.0217	0.0233

Table 2.2 shows FM of configuration 1 to 9. Note that, as $F(\varphi)$ is obtained by a time-consuming optimization, we did not build the continuous integral. We rather changed p_1 and p_1 in steps of $\Delta p_1 = \Delta p_2 = 0.001$ m (cf. Figure 2.9 - 2.17). For the optimization (2.6) we used $k_p = 1$ and $k_o = 0.05$. We simulated a mechanism with node width $w_{T/C} = 24$ mm and node height $h_{T/C} = 22$ mm. The node distance was set to $d_C = 21$ mm in case of the Cardan Element and $d_T = 19 \ldots 28$ mm in case of the Telescopic Element. These limits are realistic for a telescopic rod of that size. Joint limits for revolute joints were set to $\pm 45°$. As expected, configuration 4 and 7 deliver the most satisfying results for FM. Their design differs in the direction in which the Cardan Elements are built in. This means that the designs are similar, which explains the similar results for FM. The best result is delivered by configuration 4, which is also reflected by a flat surface in Figure 2.12 over nearly the whole parameter space. For all configurations higher F-values are gained at the boarder of the parameter space, what can be explained by reached joint limits. Peaks in the surfaces (e.g. Figure 2.14) can be explained by prematurely stopped optimizations, because each optimization was limited to 2000 iterations. As all configurations have high values at the border of the parameter space and as perfect values ($F = 0$) are reached for configuarion 4 over the rest of the parameter space, inaccuracies because of prematurely stopped optimizations can be neglected. We thus chose configuration 4 for the prototype.

2.1.5 Expanding the FO

As mentioned before, the number of DoF M must match the number of actuators a in order to gain a determined structure. Figure 2.18, 2.19 and 2.20 show three design concepts and their possible extensions, where s is the step of extension. Note that the first step ($s = 1$) describes the square grid crust with 9 nodes (Figure 2.4). It can be seen, that the FO can be arbitrarily extended to cover larger areas. Design 1 is a design where the kinematics is changing with each extension, while the resolution remains the same. A new step of extension is added to the previous step so that a mural structure is formed. By doing so, rectangular loops occur in vertical and horizontal direction whereas square loops remain in the diagonal directions (cf. Figure 2.18). Design 2 has a recursive kinematic design, so that all knowledge attained

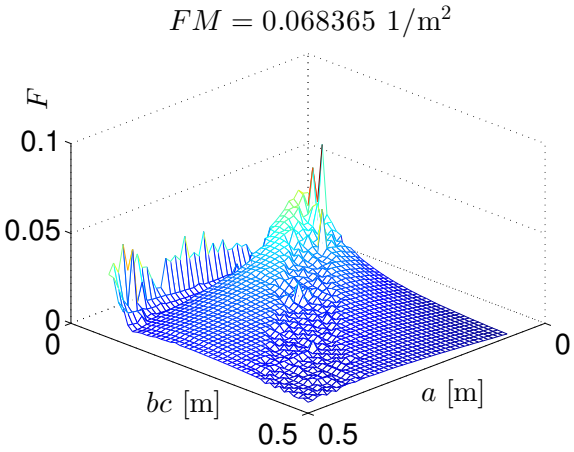


Figure 2.9: Formability of configuration 1

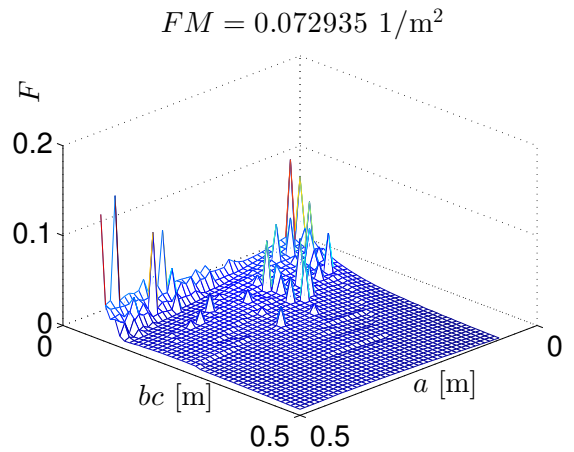


Figure 2.10: Formability of configuration 2

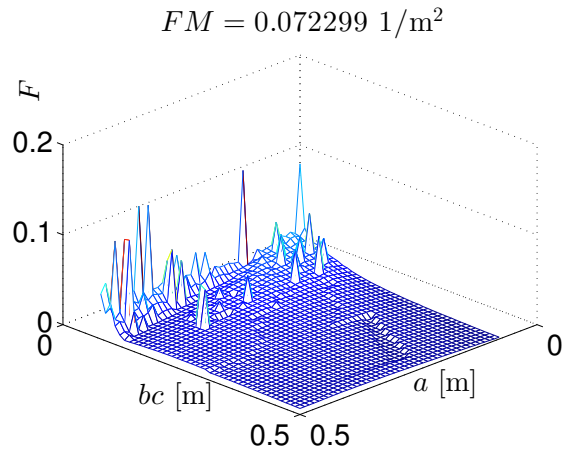


Figure 2.11: Formability of configuration 3

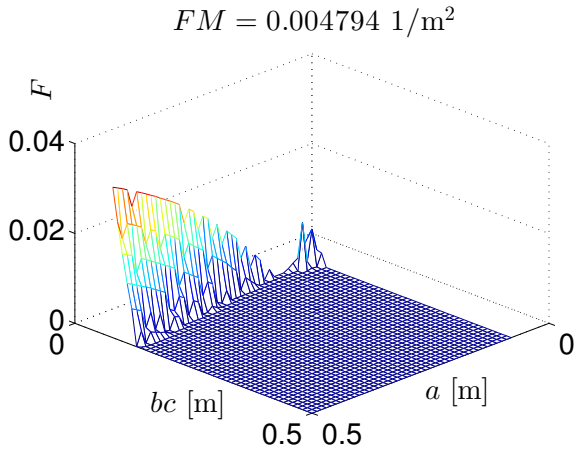


Figure 2.12: Formability of configuration 4

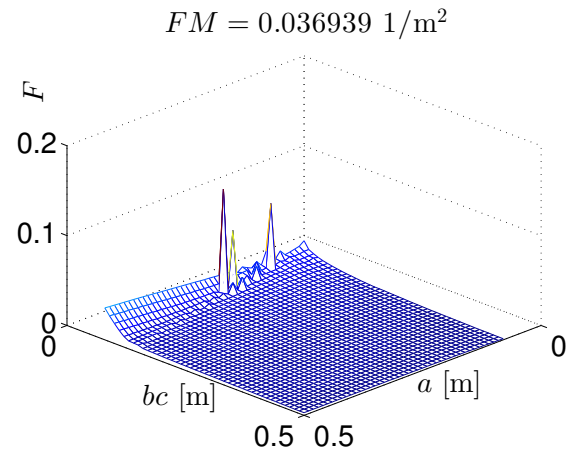


Figure 2.13: Formability of configuration 5

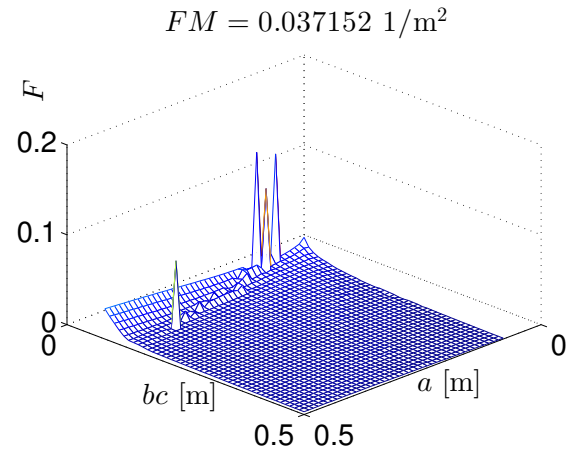


Figure 2.14: Formability of configuration 6

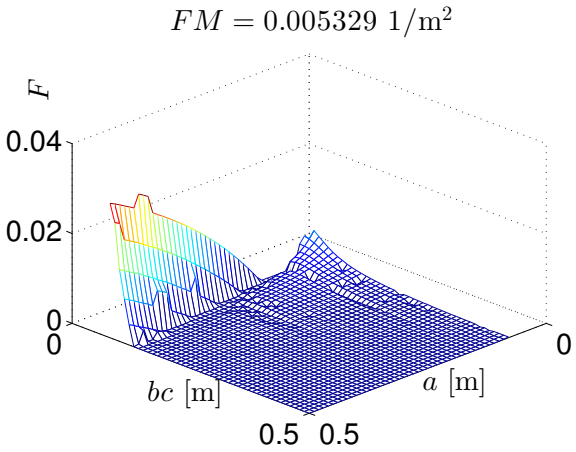


Figure 2.15: Formability of configuration 7

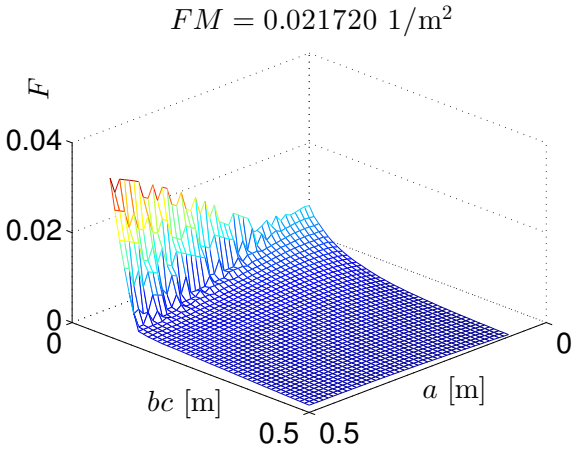


Figure 2.16: Formability of configuration 8

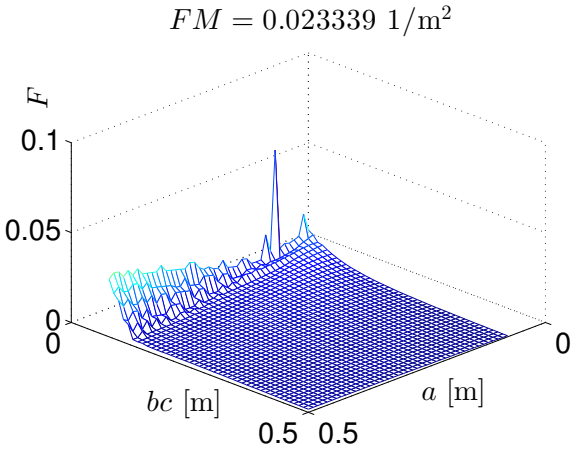


Figure 2.17: Formability of configuration 9

at step 1 can be transferred to all further steps. The recursive characteristic of design 2 becomes clearer when considering all lower steps of the kinematics as one big center node: Each new step brings in 4 new closed loops, which have the same kinematic structure as the loops of the previous steps. The drawback of the latter design, however, is that the resolution decreases with each additional layer. Similar to design 2 also design 3 has a recursive design, but with homogeneous resolution. Each step is built out of 9 elements of the previous step and is assembled in the same of the 1st step. In terms of determinacy it is not important at which nodes the single elements connect, but for simplicity connections from center node to center node between the single elements should be preferred. One can show that for each step s the number of actuators a matches the number of overall DoF, so that $a(s) \stackrel{!}{=} M(s)$ and thus, all three mechanisms are determined, for any step of extension. For the proof of determinacy please refer to (A.1), (A.2) and (A.3). Finally, it should be noted that all extension designs foresee that the FO is fixed to the ground at node 1, which means that this node has to support all interaction forces. In consequence, extension of the design is only possible up to a certain limit. The nodes, which are far away from node 1, might need extra support. Supporting the FO at more than one point would overdetermine the system, which must be respected when positioning and orienting the supported nodes in 3D space. In other words, the positions and orientations of the supported nodes are dependent on each other and must be controlled accordingly. As, however, the device is designed to be mounted at the end-effector of a robotic arm, which can move the FO in space, only areas close to the specific interaction point need to be covered. In this thesis we limit ourselves to the design and control of the basic square grid crust with 9 nodes, while all the presented methodologies also hold for the three possible extension designs.

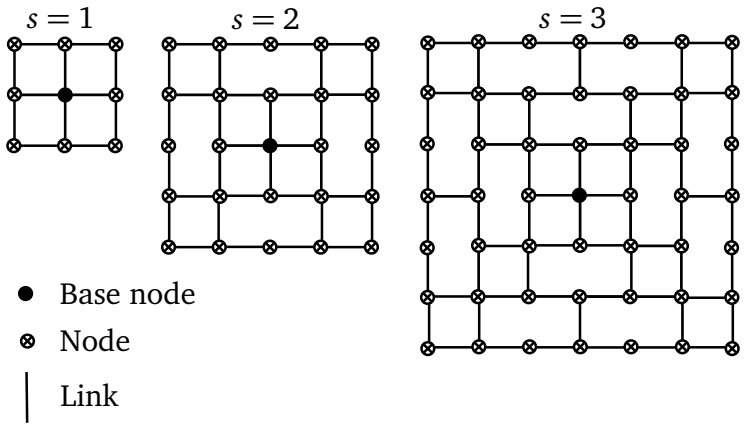


Figure 2.18: Design 1: changing kinematic design (s: step of extension)

2.2 Mechanical Construction

This section illustrates the mechanical construction of the FO considering the results of the previous section. The nodes of the FO are built from Acrylonitrile Butadiene Styrene (ABS) with the help of a rapid-prototyper. This gives us a large freedom of scope when designing the nodes. Parts, which are exposed to high mechanical stress, are made from steel, brass or

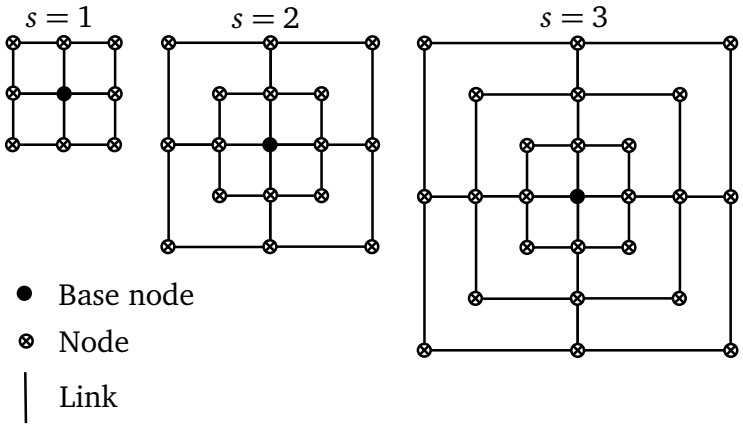


Figure 2.19: Design 2: recursive kinematic design (decreasing resolution, s: step of extension)

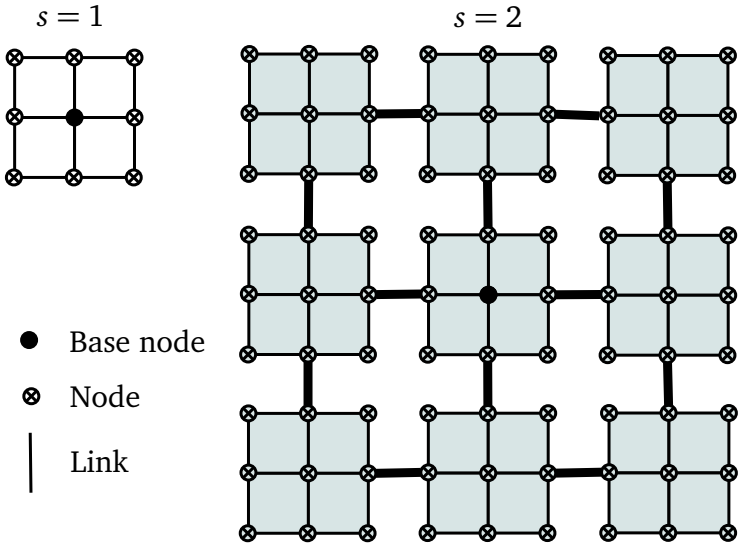


Figure 2.20: Design 3: recursive kinematic design (unchanging resolution, s: step of extension)

aluminum. As mentioned before we choose configuration 4.

2.2.1 Telescopic Element

Figure 2.21 shows the CAD construction of the Telescopic Element. The nodes (type 1) are equipped with bearing lugs, which are used to mount the telescopic rods. On these rods, which are made from brass, cable fasteners are mounted. The cable fasteners are made from ABS and are needed to fix cables, which are used to actuate the swivels at the nodes. The two DoF of the telescopic rod are unactuated and unsensed. The angles of the swivel joints are measured by magnetic encoders: A diametrically polarized circular magnet is fixed to the axis of rotation and its position is captured by a rotary position sensor (Austriamicrosystems, AS5040). The sensor has a resolution of 0.35° and the board which the sensor is attached to, is clamped to the corresponding bearing lug. The height of the nodes is $h_T = 22$ mm and the

width is $w_T = 24$ mm. The distance between two nodes is $d_T = 27$ mm ... 44 mm, depending on the configuration of the telescopic rod. An exploded view of the Telescopic Element can be found in Figure A.1.

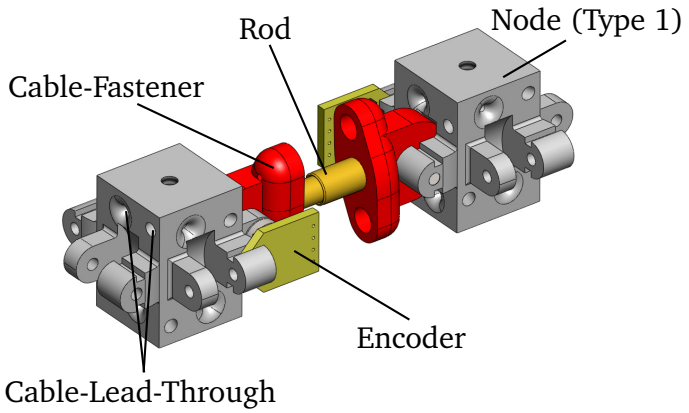


Figure 2.21: CAD drawing of Telescopic Element with 4DoF

2.2.2 Cardan Element

A CAD drawing of the Cardan Element is shown in Figure 2.22. On one side (node type 1) the Cardan Element is designed like the Telescopic Element. One sensor measures the rotational position of the swivel point at the node of type 1. The angles of the Cardan joint are difficult to measure and thus, we leave them unmeasured and capture the remaining DoF, the rotation between the rod and the node of type 2. In order to do so we designed the node so that the rod is fed through the node. The magnet, whose position is sensed by the sensor, is mounted on the other side of the node. As two rods connect to the nodes 3, 5, 7 and 9 with an angle of 90°, the jacking points are shifted upwards and downwards, to avoid an intersection of the rods. The shift S must be at least $S = D/2$, with D the diameter of the rod. In our construction the shift is $S = 1.5$mm. The dimensions of the node next to the Cardan joint (type 2) has slightly different dimensions: $w_C = 23$mm and $h_C = 25$mm. Beside the sensor, a guide washer is mounted to the rod where cables are fixed and guided for actuation purposes. An exploded view of the Cardan Element can be found in Figure A.2.

2.2.3 Formable Object

Figure 2.23a shows the CAD drawing of the FO while Figure 2.23b shows a photo of it. The center node (node 1) is extended to make the FO mountable for example to a robotic arm to increase its workspace. By doing so, the FO can be positioned and oriented in 3D space, which adds another 6 DoF to the device.

2.2.4 Motor-Module and Cable-Transmission

One big problem for shape-rendering interfaces is their actuation, because powerful actuators are not available in small scale. Putting actuators next to the joint which is to be actuated

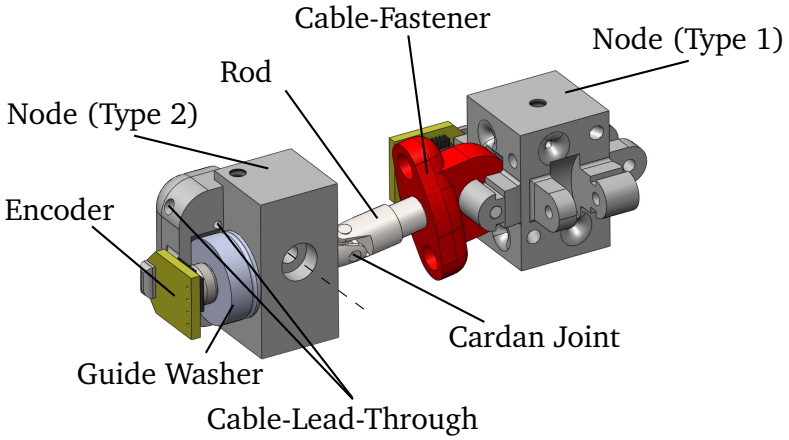
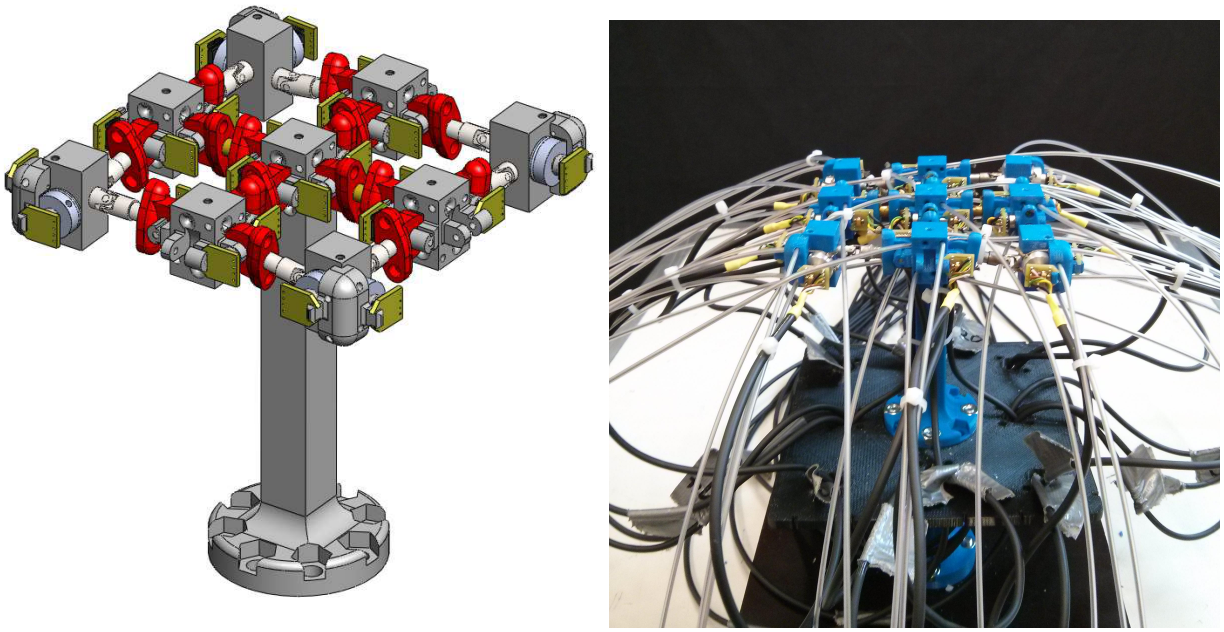


Figure 2.22: CAD drawing of Cardan Element with 4DoF



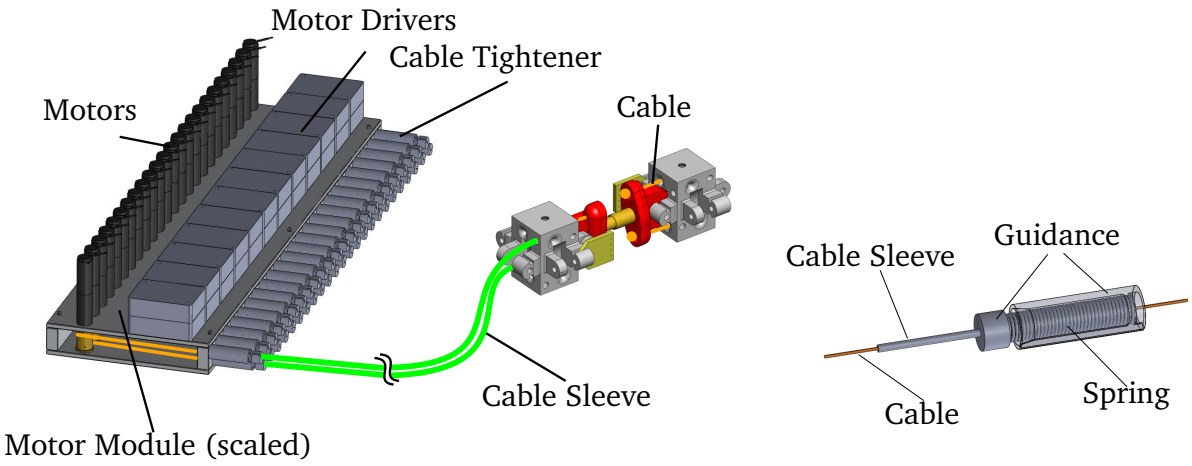
(a) CAD drawing of the Formable Object

(b) Realized prototype of the Formable Object

Figure 2.23: The Formable Object

leads to a heavy and bulky mechanism and decreases the spatial resolution of the interface. We thus, decided to use classical actuators and to transmit their power to where it is needed via cables. Figure 2.24a shows the CAD-drawing of a motor module which includes motors, motor drivers and cable tighteners. The real motor module is shown in Figure 2.25. We use Maxon's *RE-max 13* motors in combination with the planet gear *GP 13 A*, which has a reduction of 275:1. We drive the motors with the *ESCON 36/2 DC* servo controller. Cable tighteners in form of springs are used to generate pretension in the cables (Figure 2.24b). In order to drive the actuated joints in both directions the cables form a closed loop with the motor and end on both sides of the correspondent cable fastener or guiding washer, respectively. As cables we use steel wire ropes with a diameter of 0.75 mm provided by *Carl Stahl*. The cable sleeves provided by *Kavan* have a length of approximately 1m and an inner

and outer diameter of 1 mm and 2 mm, respectively. To operate the motor-drivers a PC with 3 *Mecovis IntelliDAQ MF8A*-PCI cards is used. Each card features 8 analog inputs, 8 analog outputs, 24 general purpose I/Os, 24 digital ports and 8 quadrature encoders.



(a) CAD-drawing of motor module and basic element

(b) CAD-drawing of cable tightener

Figure 2.24: The motor module of the Formable Object

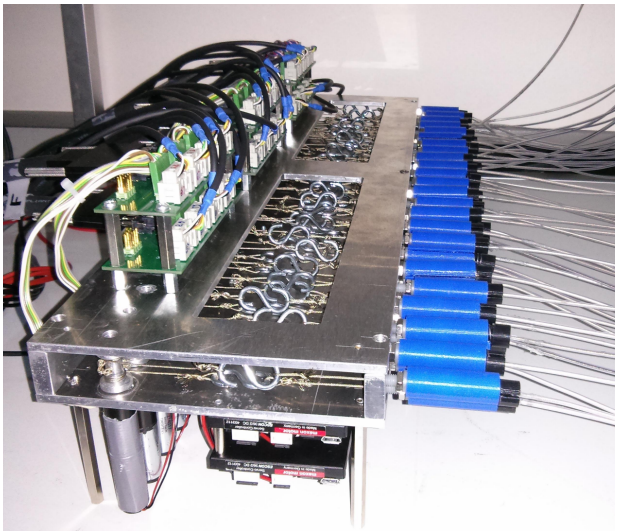


Figure 2.25: Prototype of the motor module

2.3 Control

The overall control scheme to drive the FO is shown in Figure 2.26. We assume that desired joint angles are either calculated offline using an optimization function (2.6) or a differential kinematics method ([41] and Chapter 3). The inverse kinematics delivers desired joint angles of active joints φ_a, which can be measured by sensors and actuated by motors via the cable transmission. As the passive joints φ_p are dependent on the active joints they are

not used for the control of the FO. The active joints φ_a are controlled using a classical PID-controller. Note that the PID controller does not directly receive the error of the active joints, but that the error is first fed through a dead zone block, which eliminates errors smaller than 0.4 deg. This is important to avoid permanent movements in the region of the desired values. The output of the PID controller is the motor current $\boldsymbol{i}$.

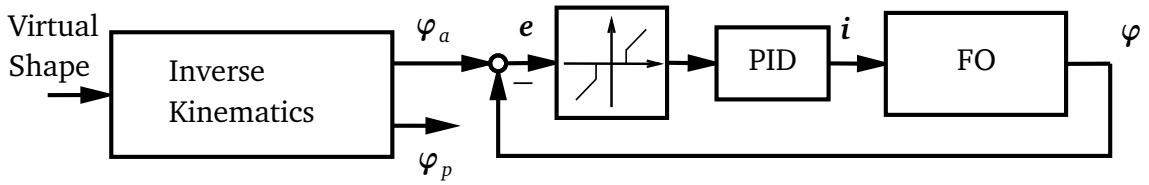


Figure 2.26: Signal flow to control the FO

2.4 Evaluation

2.4.1 Rendering of Different Curvatures

In this section, the mechanical design is evaluated with respect to its capability of rendering different curvatures using the Formability Measure. At first, we performed similar simulations as in Section 2.1.4, but now considering the construction-conditioned changes. These changes are different node dimensions for type 1 and 2 and the previously mentioned shift of the jacking point of the rod of the Cardan joint. Figure 2.27 shows the formability of the real design over the parameter-space introduced earlier. The Formability Measure is $FM = 0.0050\ 1/\mathrm{m}^2$.

Next, the maximal curvatures, which can be rendered with the kinematics, are analyzed. In [39] we investigated the ability of configuration 1 to render cylindrical and spherical shapes. We further distinguished between concave and convex shapes. Worse results were obtained in case of spherical shapes whereas concave shapes could be rendered better than convex shapes. This behavior is also reflected in Figure 2.9. To analyze the kinematics of the FO we proceed as detailed in [39]: We stepwise (1mm steps in radius) change the curvature from 0 (flat) to higher values (cylindrical/spherical) and check at each step whether the shape was rendered or not. We define a shape to be 'rendered' when all position errors are below 1mm and all orientation errors are below 1°. More complex shapes than cylinders or spheres are not considered as they can mostly be composed of a series of basic convex and concave shapes and the device is supposed to only render the local surface area, which is close to the interaction point. Rendering of surfaces with finer details would require far more than 9 nodes and also a higher resolution. Table 2.3 shows the maximum reachable curvatures $\rho = 1/R$ of the FO.

Figure 2.28a and 2.28b show the FO rendering parts of a cylinder surface with radius $R = 0.08$ m. Figure 2.28c shows the FO rendering part of a sphere surface with radius $R = 0.1$ m.

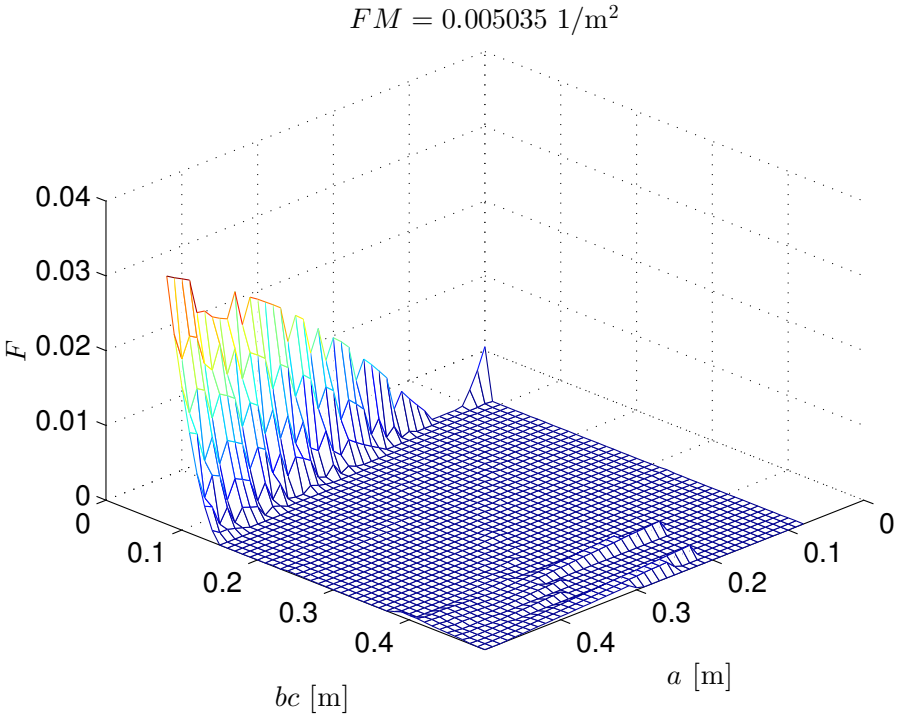


Figure 2.27: Formability of the real configuration

Tabular 2.3: Maximum curvatures of Formable Object

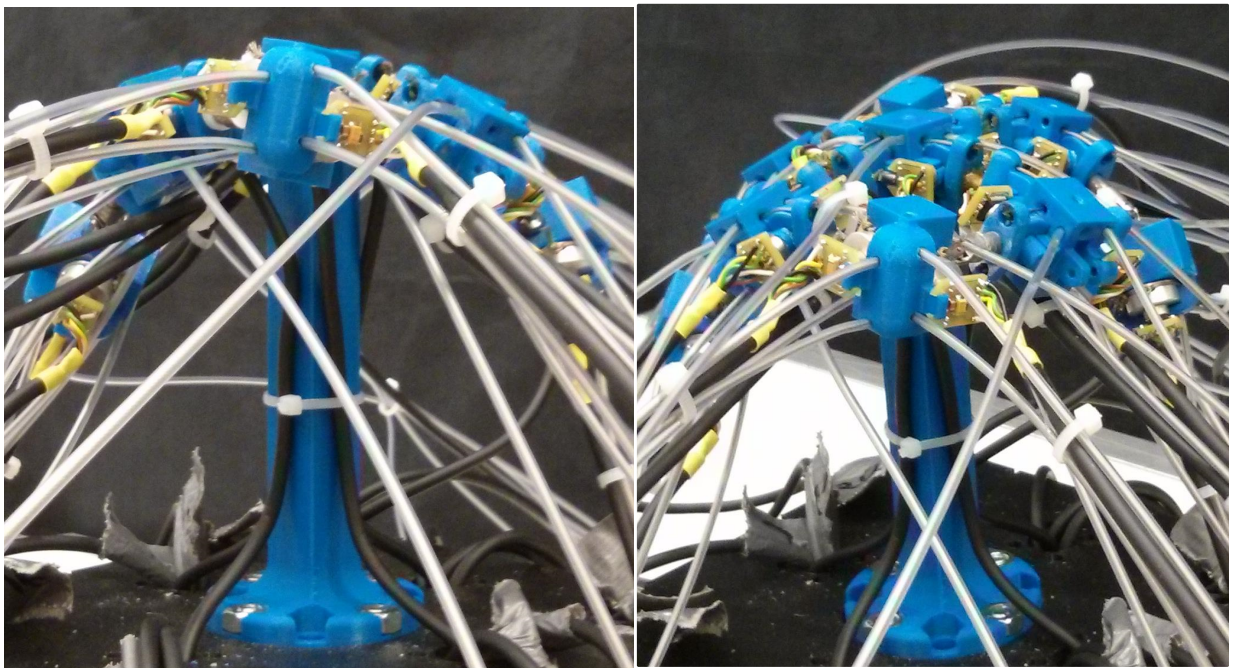
	Concave	Convex
Spherical	$16.67\frac{1}{\mathrm{m}}$	$11.36\frac{1}{\mathrm{m}}$
Cylindrical	$71.43\frac{1}{\mathrm{m}}$	$15.15\frac{1}{\mathrm{m}}$

2.4.2 Stiffness

The stiffness of the kinematics was determined by applying normal forces to the nodes and measuring the corresponding offset from the initial position. This was achieved by attaching a spring scale to the nodes, see Figure 2.29, and using the built-in sensors to read the current configuration of the FO. The Cartesian position of the nodes is calculated with the help of the forward kinematics. The forward kinematics of the FO is calculated offline by solving an optimization problem analogous to the inverse kinematics problem discussed in [39] or 2.1.4, respectively. We therefore distinguish between unknown/passive joint angles φ_p and measurable/active joint angles φ_a. The loops $L_1 \dots L_4$, can be expressed as homogeneous transformations $\boldsymbol{T}_1(\varphi_p, \varphi_a) \dots \boldsymbol{T}_4(\varphi_p, \varphi_a)$ and when the loop constraints are satisfied $\boldsymbol{T}_j = I$ must hold (see (2.2)-(2.5)). Thus, in order to find the solution of the forward kinematics one has to minimize $\boldsymbol{T}_j - \boldsymbol{I}$ for all 4 loops:

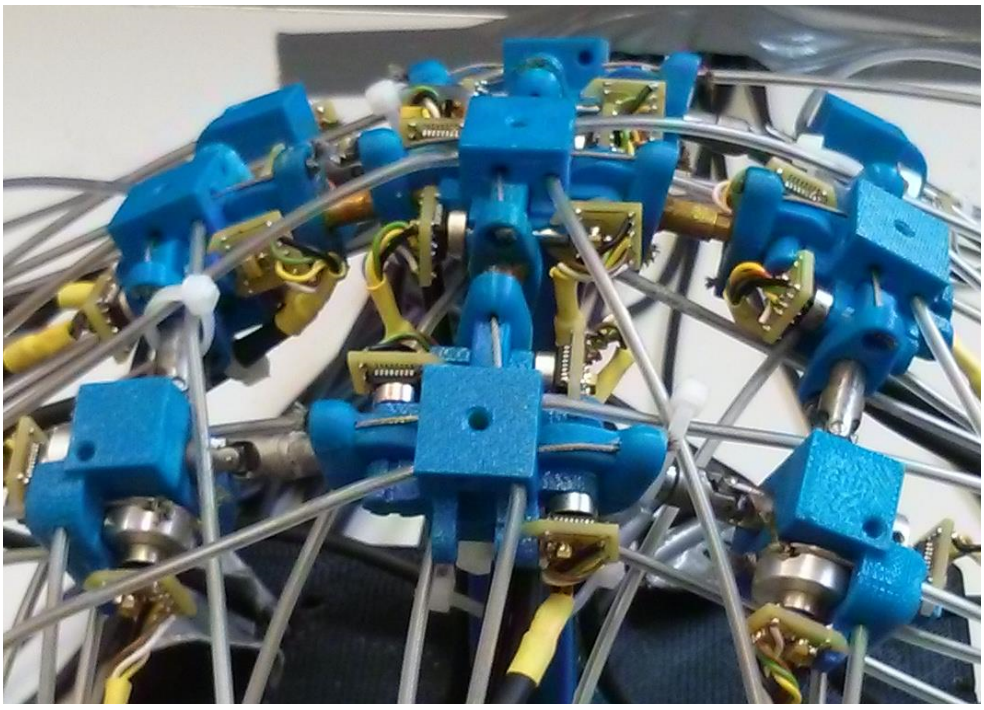
$$\min_{\varphi_p} \quad \tilde{F}(\varphi_p, \varphi_a) = \sum_{j=1}^{4} (\boldsymbol{T}_j(\varphi_p, \varphi_a) - \boldsymbol{I}) \tag{2.10}$$

Note that, by using the forward kinematics, node displacements created by the deformation of the structure or by backlash in the joints, cannot be observed. Thus, a more accurate



(a) Part of a cylinder surface (view 1)

(b) Part of a cylinder surface (view 2)



(c) Part of a sphere surface

Figure 2.28: The FO rendering surfaces

method would be to measure node displacements by an external measurement device. As the major reason for node displacement is backlash in the cable and the cable tighteners, we settled for using this measurement method. Once the passive joint angles φ_p are calculated the Cartesian positions of the nodes can be derived analytically. Thus, the Cartesian deflection of the nodes can be calculated from the measurable joint angles φ_a.

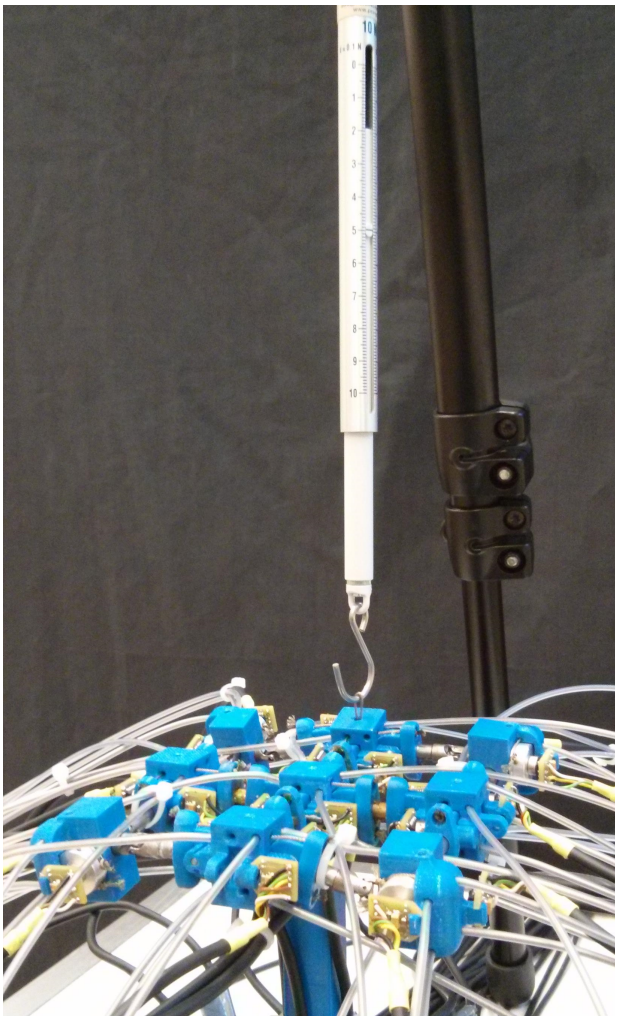


Figure 2.29: Determing the stiffness of the FO

Uncontrolled Device

Measurements were performed with disabled motors. By doing so, the displacement of the nodes is only dependent on the elasticity of the mechanism (mainly elasticity of cable pulls and the pre-tensioning elements). Figure 2.30 and 2.31 show results for nodes of type 1 (2, 4, 6 and 8) and nodes of type 2 (3, 5, 7 and 9). As the path length to the center node (node 1) for type 2 is longer, we expect a lower stiffness for these nodes compared to the nodes of type 1. The force was increased and decreased in steps of 1N. For each node, one series of measurement was performed and the mean was calculated from the series for type 1 and type 2 nodes, respectively. For the nodes of type 1 we determined a stiffness (in

force direction) of 2.3 N/mm for increasing forces and a stiffness of 12 N/mm for decreasing forces. For the nodes of type 2 we determined a stiffness (in force direction) of 0.68 N/mm for increasing forces and a stiffness of 3.4 N/mm for decreasing forces. It can be seen that when returning to the initial point ($F = 0$ N), a significant deflection (2.6 mm and 8.3 mm) remains. This deflection can be traced back to friction in the joints of the kinematics and in the cable pulls.

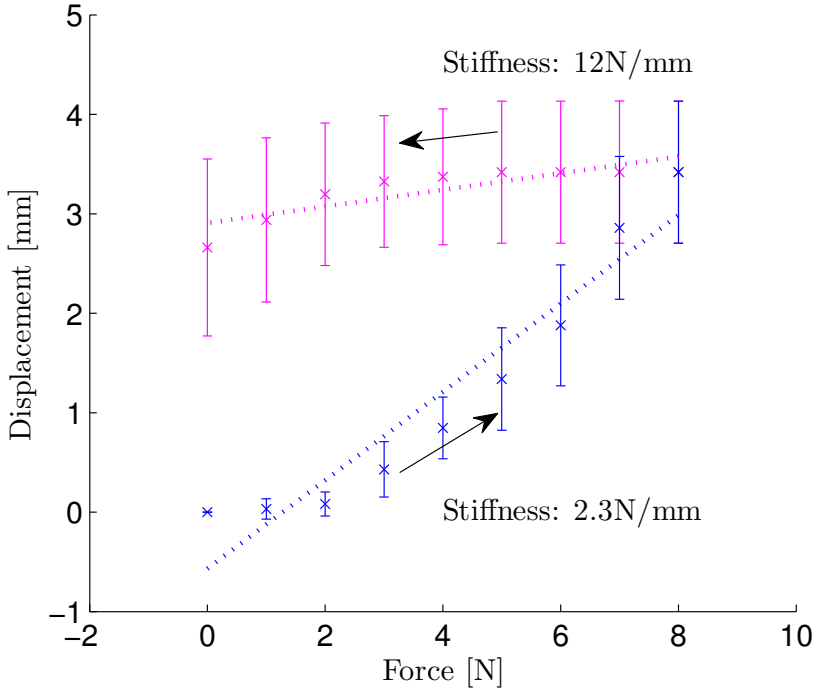


Figure 2.30: Maximum stiffness of FO when applying normal forces at nodes 2, 4, 6 and 8 (unpowered)

Controlled Device

Analog measurements were performed for the kinematics with enabled motors/control. Higher stiffness and no hysteresis is expected compared to the measurements performed with disabled motors. As can be seen in Figure 2.32 and 2.33 significant higher stiffness values were determined with enabled control. For nodes of type 1 we determined a stiffness of 9.6 N/mm while for type 2 the stiffness was found to be 4 N/mm. In the actuated case no significant difference was observed for increasing and decreasing forces.

2.4.3 Dynamic performance

In this section the dynamic performance of the FO is analyzed by switching between desired joint angles for a flat configuration and for a configuration rendering a cylinder with $R = 0.1$ m. By measuring the actual joint angles over time the node errors (position $\boldsymbol{e}_p$ and orientation $\boldsymbol{e}_o$) of all nodes can be calculated using (2.10). Figure 2.34 shows the maximum position and orientation error of all nodes. Figure 2.35 shows the mean position and orientation error of all nodes. One can see that it takes approximately 2s until a shape is rendered

after a switch. When a shape is rendered, a remaining error can be observed. This remaining error can be explained by a) saturated actuators, b) the finite resolution of sensors, c) the dead zone in the control loop (c.f. Figure 2.26), which reduces stick-slip effects in the region of the desired values but does not allow accurate control.

2.5 Conclusion

This chapter introduced the design of a SRI with 9 surface points and 24 DoF, which is fully actuated. Based on a preliminary design version an improved optimal design was found and the improvement was quantitatively illustrated with simulations. Details about the hardware design, actuation and sensing were given. Possibilities to extend the area of interaction of the FO were discussed. Two possibilities with a recursive kinematic design and one possibility with changing kinematic design were presented. Maximum curvatures allowed to render cylinders and spheres were determined. Furthermore, the stiffness of the kinematics was measured in a flat configuration. When the control of the FO was enabled we reached a stiffness of > 4 kN/m which is above the demanded 1 kN/m. When switching between shapes, we found that approximately 2s are needed until the new shape is rendered. When rendering a shape a remaining shape error was observed which can be explained by saturated actuators and inaccuracies in the control of the joint angles.
Future work must target a design, which keeps the configuration of the mechanism, but is optimized to reduce friction so that a more accurate control of the active joint angles can be achieved. Figures A.3 - A.6 illustrate how, by redesigning the motor module and the FO, shorter cables length and better installations of the cables can reduce the friction.

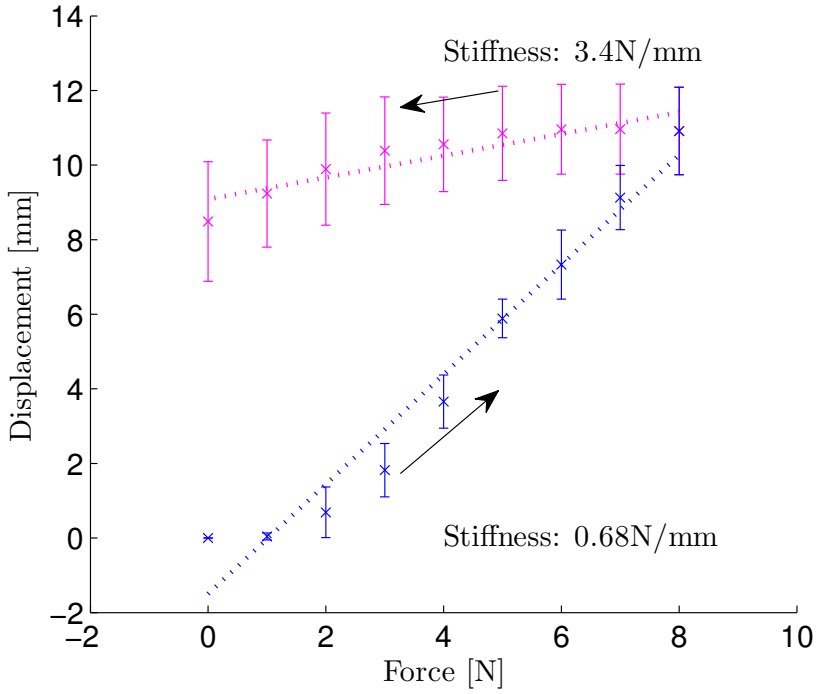


Figure 2.31: Maximum stiffness of FO when applying normal forces at nodes 3, 5, 7 and 9 (unpowered)

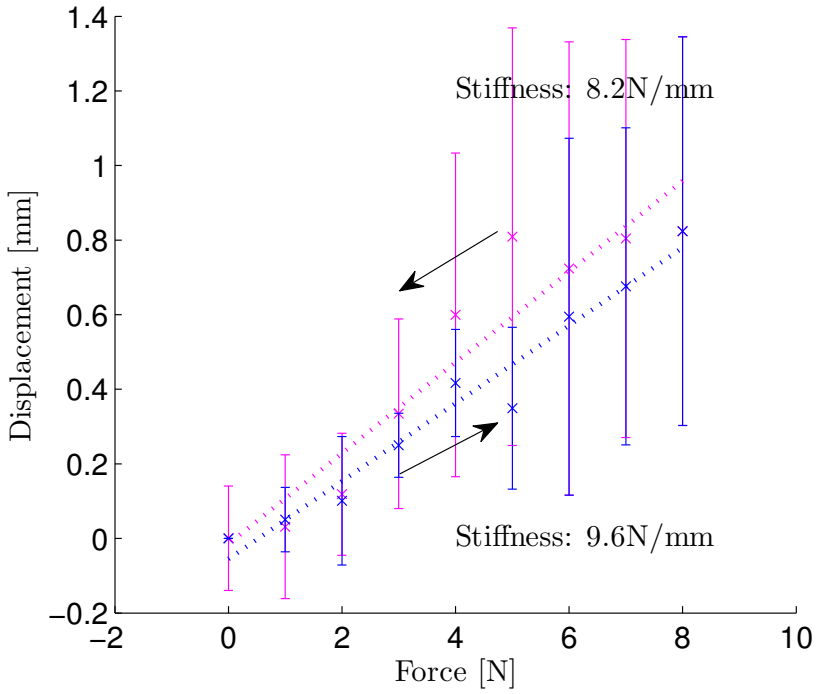


Figure 2.32: Maximum stiffness of FO when applying normal forces at nodes 2, 4, 6 and 8 (powered)

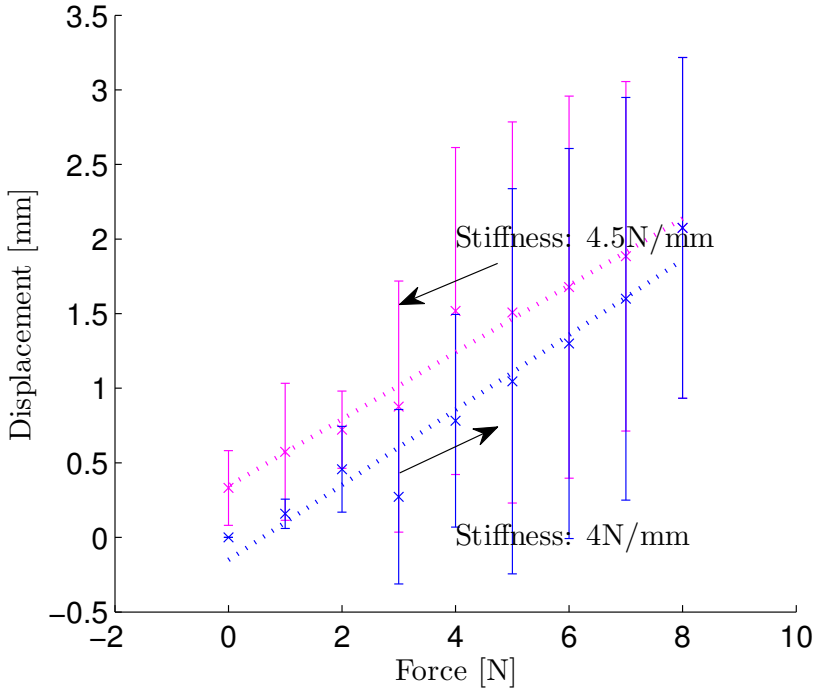


Figure 2.33: Maximum stiffness of FO when applying normal forces at nodes 3, 5, 7 and 9 (powered)

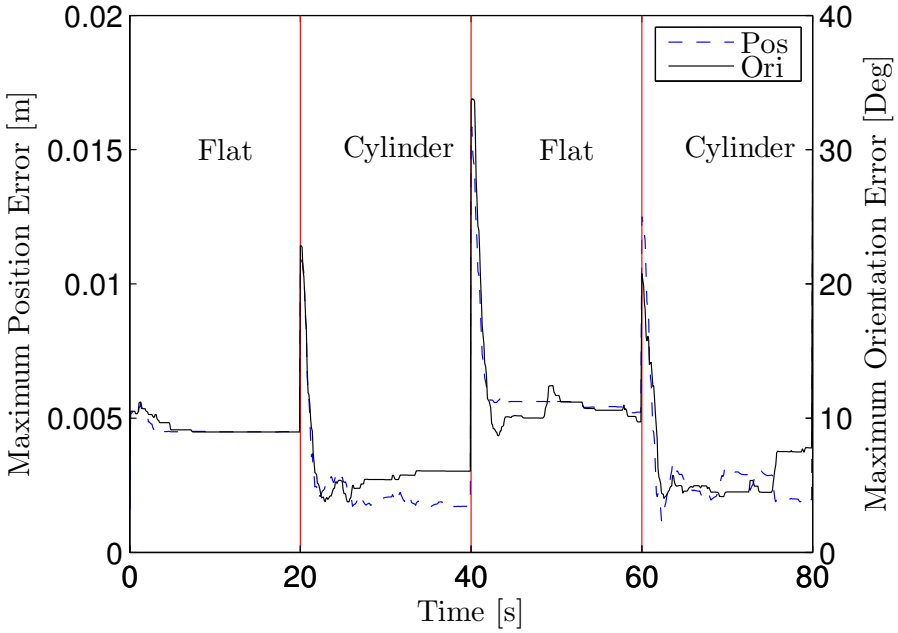


Figure 2.34: Maximum errors (position & orientation) of all 9 nodes

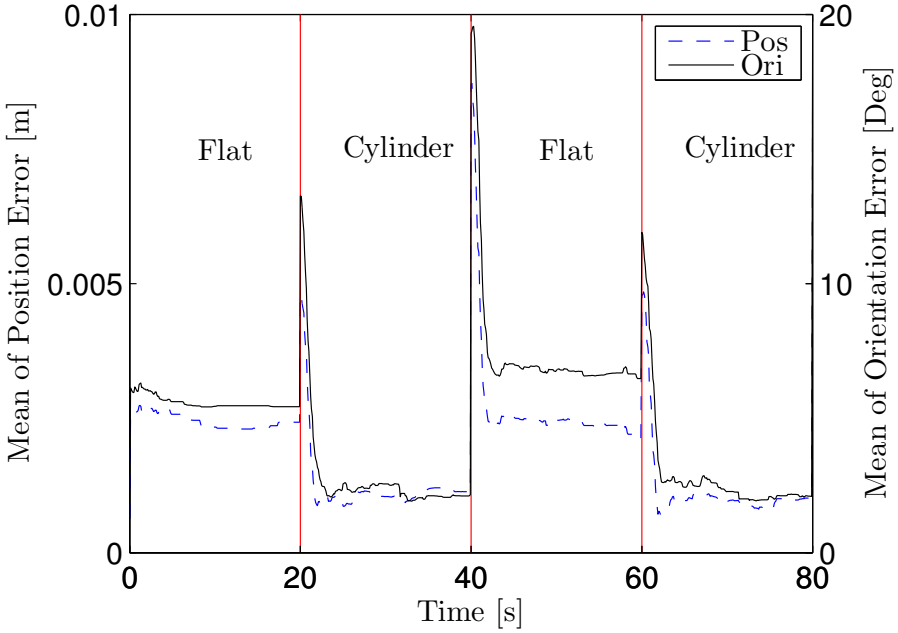


Figure 2.35: Mean of errors (position & orientation) of all 9 nodes

3 Inverse Kinematics for Shape Rendering Interfaces

Dynamically controlling the shape of shape rendering interfaces (SRI) is a major challenge when dealing with them, especially when they have a parallel kinematic design like the FO presented in the previous chapter. In a classical sense, the inverse kinematics of a robot maps the Cartesian position of the end-effector to the joint angles (generalized coordinates) of the mechanism. In the case of SRIs, inverse kinematics must be defined in a different way, as we are not dealing with one end-effector, but with solid bodies within the mechanism, which are used to approximate a shape of interest. How to position and orient these solid bodies must be analyzed first. SRIs, in most cases, feature a parallel kinematic design as parallel kinematics feature high stiffness and speed. SRIs are, in most cases, highly redundant, which further complicates the inverse kinematics problem.
In [44], methods to derive the inverse kinematics for parallel kinematics are grouped into *analytical methods* and *geometrical methods*. Within both methods, a system of equations is generated. Geometrical methods in contrast to analytical methods capitalize on the fact, that for each leg of a parallel mechanism the extremities have a known position in 3D space. Cutting these legs and calculating intersections of the varieties of the two sides of the cut leads to a system of equations of which the solution is easy in most cases. Both methods, however, cannot be calculated in real time with a sampling rate of more than 1kHz as typically required in haptic applications due to the complex parallel kinematics with highly nonlinear equations. Common parallel kinematics for which analytical/geometrical methods exist and which are simple enough to be calculated in real-time are the Delta robot [45] or the Stewart platform [46].
In [47] an optimization-based method is presented, where the volume between the shape rendering interface and a virtual object is minimized. The optimization is based on a so called physically-based modeling technique where the interface is physically modeled as a mass-spring system and contact forces, which deform this model, are generated by bringing a model of the object to render in contact with the model of the interface. By doing so, update rates of 11.5Hz were obtained, which is not sufficient for haptic applications where update rates of at least 1kHz are needed to ensure a passive system, which remains stable even when stiff objects are rendered (cf. [48]). In [39] we followed a similar approach (analytical method), based on a constrained optimization method (cf. (2.6)), but we also found this implementation unsatisfying because of the rather long converging times of the numerical optimization (30 s $-$ 2 min).
On the other hand, inverse kinematics of redundant serial manipulators can easily be solved in real time using a differential kinematics approach. A hierarchy is formulated and tasks of lower priority are projected into the Nullspace of respective higher order tasks [24] [25] [26]. In this context tasks refer to the control of the position and/or orientation of the end-effector or points on the kinematic structure.

In this chapter we present an approach of controlling the shape of 3D parallel kinematics. The proposed method is real-time capable, based on a differential kinematics approach and allows to render shapes by simultaneously taking into account loop constraints, joint limits and user interaction points. A systematic approach for solving the inverse kinematics of SRIs that consist of multiple nodes is derived. These nodes are to be positioned on and oriented to a virtual surface. We therefore reinterpret the differential kinematics approaches for serial kinematics to use them for SRIs, which feature a parallel kinematic design. To control the shape, differential kinematics are used within a hierarchical framework. Shape- and loop-constraints and user interaction are considered and joint limitations are taken into account. In [41] we already presented preliminary considerations. Further, two alternative control modes for stiff and compliant objects are presented. Finally, two methods for proximity queries are specified: Implicit surfaces as well as polygon sets can be used as input for the virtual shape, which is to be rendered.

3.1 Problem Description

To define the inverse kinematics problem, a general specification of the mechanism we are dealing with is needed. Our parallel kinematics consists of nodes and connectors between these nodes. The nodes are covered by an elastic layer and are used to approximate the desired shape. Each node represents a small area of the virtual surface and has a body-fixed coordinate system defining its position $\boldsymbol{p}$ and orientation (cf. Figure 3.1). The point $\boldsymbol{p}$ is located on the nodes' surface which is used to render the virtual shape. The nodes are linked by connectors, which can have several joints. In general only a few joints are actuated (active joints) whereas the others remain unactuated (passive joints), in order to gain a determined system. At this stage, however, it is not important which joints are actuated, as the proposed algorithm does not depend on this selection. We assume that we are dealing with a determined system and the passive joints adapt to the active joints as they are dependent on them. We further assume that a proper selection was made for the actuated joints within the design of the SRI. The kinematics has M DoF in total. In the following, we analyze aspects, which have to be considered in shape rendering.

3.1.1 Aspects of Shape Rendering

When rendering shapes with SRIs, multiple aspects have to be considered simultaneously. These are *shape forming*, *loop constraints*, *joint limits* and *user interaction*.

Shape Forming

We assume that a virtual surface to be displayed by the mechanism is given by a 3D implicit function $f(x,y,z)$ or a polygon set (triangles). To render the shape of the virtual surface the nodes have to be positioned on the surface and aligned normal to it.

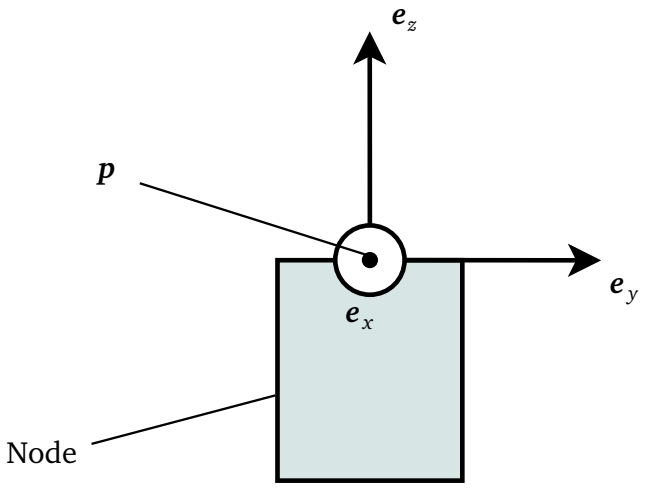


Figure 3.1: Node with body-fixed coordinate system

Loop Constraints

The mechanism is constrained by a certain number of loops l which form closed kinematic chains between the m nodes of the mechanism (see Figure 3.2). When rendering shapes, the corresponding loop constraints must hold at every time. Homogeneous transformations ${}^{j}\boldsymbol{T}_{i}$ can be used to mathematically describe the loop constraints and the relative position and orientation between nodes i and j, respectively. In Figure 3.2 a mechanism with $m = 10$ nodes is shown. The loop u contains m_u nodes. For each loop a directed graph is introduced with vertices $v_1, \cdots, v_{m_u+1}$. With this notation, the closed loop constraints can be described as follows:

$$L_u : \prod_{t=1}^{m_u} ({}^{v_t}\boldsymbol{T}_{v_{t+1}}) - \boldsymbol{I} = \boldsymbol{0}, \qquad \text{with: } u = 1 \ldots l \tag{3.1}$$

This means that, when transforming from one body-fixed coordinate system to another successively, no displacement is observable when closing the loop (${}^{i}\boldsymbol{T}_{i} = \boldsymbol{I}$). For the example in Figure 3.2 the first loop constraint is given by:

$$L_1 : {}^{1}\boldsymbol{T}_{2} \cdot {}^{2}\boldsymbol{T}_{9} \cdot {}^{9}\boldsymbol{T}_{10} \cdot {}^{10}\boldsymbol{T}_{7} \cdot {}^{7}\boldsymbol{T}_{8} \cdot {}^{8}\boldsymbol{T}_{1} - \boldsymbol{I} = \boldsymbol{0} \tag{3.2}$$

Each loop provides 6 independent equations (3 equations for the position and 3 equations for the orientation). This means that $6l$ equations have to be satisfied to meet the given constraints. Free nodes can be positioned and oriented within these constraints, using the redundancy of the kinematics.

Joint Limits

Beside loop constraints, joint limits have to be considered. The joints can be prismatic or rotational and can be moved within their physical limits only.

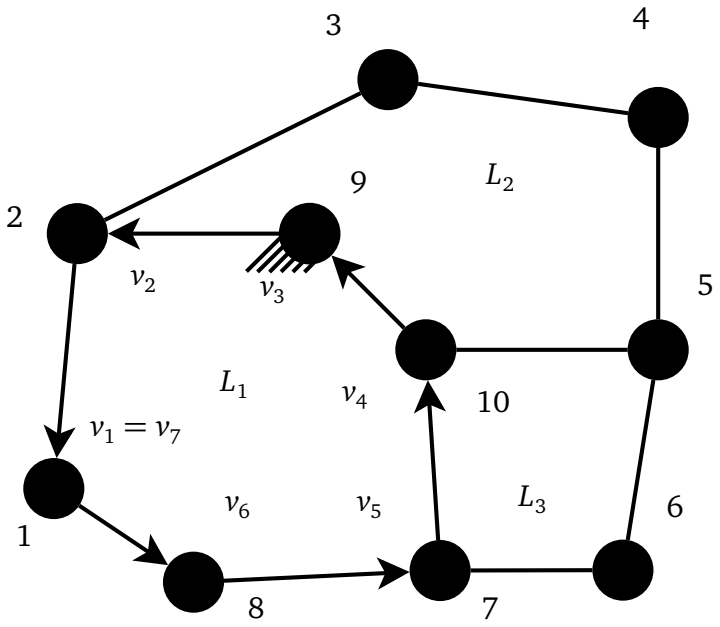


Figure 3.2: Mechanism with 10 nodes and three closed loops with base node 9 fixed to the environment

User Interaction

When rendering shapes, different types of user interactions have to be considered, especially when compliant objects should be rendered. During haptic interaction, node movements under the elastic layer should not generate unrealistic haptic impressions (shear-forces).

3.2 Shape Rendering

The above described aspects can be taken into account by formulating the problem of shape rendering as a nonlinear constrained optimization problem (cf. Chapter 2.1.4). Such an optimization, however, is very time consuming and the time to converge cannot be predicted. Thus, the optimization cannot run in real time with sampling rates of about 1 kHz as typically required for haptic applications. In the following sections we define the above problem as a differential kinematics problem using multiple tasks ordered in a hierarchical way. By doing so, the problem is formulated symbolically and every iteration requires the same number of calculations.

3.2.1 Hierarchical Nullspace Projection

In [24] an algorithm to solve multiple tasks in a hierarchical way was presented. Our approach is an extension of [24], as it is used to derive the inverse kinematics for SRIs, which feature a parallel kinematic design. This section briefly summarizes the main ideas of this work before introducing its extension to shape rendering interfaces. In a first step, the different tasks have to be defined. The tasks can take care of various aspects. In most cases, a task is to understand as the holding of a specific Cartesian position or a distance/orientation

relative to an object. When multiple tasks are defined, their importance is reflected in the hierarchy of the algorithm. Tasks of higher importance are solved accurately while tasks of lower importance are solved as good as possible, without influencing the higher priority tasks. This is done by a projection of the tasks on the Nullspace of the respective higher priority task. This means that higher order tasks are solved using all DoF of the mechanism, while lower order tasks use the redundancy of the mechanism without affecting the higher order tasks. The Nullspace of a task k can be calculated as follows [49]:

$$\boldsymbol{N}_k = \boldsymbol{I} - \boldsymbol{J}_k^* \boldsymbol{J}_k, \tag{3.3}$$

with $\boldsymbol{J}_k$ the Jacobian of the task and $\boldsymbol{J}_k^*$ its weighted-damped Pseudoinverse

$$\boldsymbol{J}_k^* = \boldsymbol{W}_k^{-1} \boldsymbol{J}_k^T (\boldsymbol{J}_k \boldsymbol{W}_k^{-1} \boldsymbol{J}_k^T + \lambda_k^2 \boldsymbol{I})^{-1} \tag{3.4}$$

with weighting matrix $\boldsymbol{W}$ and damping factor λ. Details about these parameters will be given later. In the following equations the algorithm [24] is shown for multiple tasks where task $k-1$ has higher priority over task k:

$$\begin{aligned}
&\underline{\text{Task 1:}}\ \dot{\boldsymbol{\varphi}}_1 = \boldsymbol{J}_{1N}^* \hat{\dot{\boldsymbol{x}}}_1 \\
&\quad \text{with: } \boldsymbol{J}_{1N} = \boldsymbol{J}_1 \\
&\qquad\quad \hat{\dot{\boldsymbol{x}}}_1 = \dot{\boldsymbol{x}}_1 \\
&\underline{\text{Task 2:}}\ \dot{\boldsymbol{\varphi}}_2 = \boldsymbol{J}_{2N}^* \hat{\dot{\boldsymbol{x}}}_2 \\
&\quad \text{with: } \boldsymbol{J}_{2N} = \boldsymbol{J}_2 \boldsymbol{N}_1 \\
&\qquad\quad \boldsymbol{N}_1 = \boldsymbol{I} - \boldsymbol{J}_1^* \boldsymbol{J}_1 \\
&\qquad\quad \hat{\dot{\boldsymbol{x}}}_2 = \dot{\boldsymbol{x}}_2 - \boldsymbol{J}_2 (\boldsymbol{J}_1^* \dot{\boldsymbol{x}}_1) \\
&\qquad\quad \vdots \\
&\underline{\text{Task K:}}\ \dot{\boldsymbol{\varphi}}_K = \boldsymbol{J}_{KN}^* \hat{\dot{\boldsymbol{x}}}_K \\
&\quad \text{with: } \boldsymbol{J}_{KN} = \boldsymbol{J}_K \boldsymbol{N}_{K-1} \\
&\qquad\quad \boldsymbol{N}_{K-1} = \boldsymbol{N}_{K-2} \left[\boldsymbol{I} - (\boldsymbol{J}_{(K-1)N}^* \boldsymbol{J}_{(K-1)N}) \right] \\
&\qquad\quad \hat{\dot{\boldsymbol{x}}}_K = \dot{\boldsymbol{x}}_K - \boldsymbol{J}_K \sum_{k=1}^{K-1} (\boldsymbol{J}_{kN}^* \hat{\dot{\boldsymbol{x}}}_k)
\end{aligned} \tag{3.5}$$

In (3.5), a task is formulated in terms of task space velocities $\dot{\boldsymbol{x}}$ (e.g. Euclidean space) while $\hat{\dot{\boldsymbol{x}}}$ denotes task velocities which also consider velocities caused by higher order tasks. The joint velocities to be sent to the low-level controllers are given by:

$$\dot{\boldsymbol{\varphi}} = \dot{\boldsymbol{\varphi}}_1 + \dot{\boldsymbol{\varphi}}_2 + \ldots + \dot{\boldsymbol{\varphi}}_K \tag{3.6}$$

3.2.2 Desired Task Velocities

In order to apply the algorithm described in section 3.2.1 to the shape rendering device, the speed for the corresponding tasks $\dot{\boldsymbol{x}}_k$ is needed. The speed $\dot{\boldsymbol{x}}_k$ is defined as the desired speed of the task $\dot{\boldsymbol{x}}_{kd}$. An error term is needed to compensate inaccuracies in the velocity originating

from the weighted-damped Pseudoinverse $\boldsymbol{J}_k^*$ or rather from $\boldsymbol{W}_k$ and λ_k. As the weighted-damped Pseudoinverse does not represent the exact inverse of the Jacobian, erroneous joint angle velocities $\dot{\boldsymbol{\varphi}}_k$ are calculated in (3.5), especially when the mechanism is close to a joint limit or singularity. Thus, the task velocity is calculated as:

$$\dot{\boldsymbol{x}}_k = \dot{\boldsymbol{x}}_{kd} + K_k \boldsymbol{e}_k = \dot{\boldsymbol{x}}_{kd} + K_k(\boldsymbol{x}_{kd} - \boldsymbol{x}_k). \tag{3.7}$$

Note that the gain matrix K_k can be split into $K_k = \begin{bmatrix} K_{kp} & K_{ko} \end{bmatrix}^T$ with different gains for positions and orientations. This formulation (3.7) requires the selection of desired positions and orientations $\boldsymbol{x}_{kd}$ as well as the calculation of actual positions and orientations $\boldsymbol{x}_k$ of the single nodes by means of the forward kinematics.

3.2.3 Task Definition

In section 3.2.1 we generally introduced the hierarchical Nullspace projection. In the following, we define tasks which need to be fulfilled for shape rendering. In the remainder of this work we consider a Node Positioning task which will be identified with index 'P', a Shape Forming task denoted with 'S' and a Loop Constraint task identified with 'C', where 'C' takes priority over 'P' and 'S' and 'P' takes priority over 'S'. With this indexing $\dot{\boldsymbol{\varphi}}$ becomes:

$$\dot{\boldsymbol{\varphi}} = \dot{\boldsymbol{\varphi}}_C + \dot{\boldsymbol{\varphi}}_P + \dot{\boldsymbol{\varphi}}_S \tag{3.8}$$

Note that joint angles $\boldsymbol{\varphi}$ are obtained by integrating (3.8). The constraint task 'C' has highest priority because these (physical) constraints must hold at every time. In the Node Positioning task 'P', specific nodes are positioned at their desired positions in Cartesian space. To do that, the redundancy of task 'C' is used as task 'P' is executed on the Nullspace of task 'C'. In the Shape Forming task 'S', nodes (which were not considered in 'P') are not placed at exact Cartesian positions, but may take multiple positions, which render the virtual shape. By doing so less redundant DoF of the higher order tasks 'C' and 'P' are needed. Note that task 'P' is only used when haptic interaction takes place with a compliant object. In this case, some nodes need to be positioned at exact Cartesian positions to avoid unrealistic haptic impressions. If stiff objects are rendered and its shape does not change over time, task 'P' is not used at all and task 'S' is on the 2nd level of the hierarchy after task 'C' as visualized in Figure 3.3. The three tasks 'C', 'P' and 'S' will be mathematically defined in the following sections.

3.2.4 Loop Constraint Task

The loop constraints of the parallel kinematics are described in a differential way. The loops are therefore virtually cut into open loops at arbitrary positions. Figure 3.4 shows the previous example in an open loop configuration. Thus, the kinematic chains can be considered as serial kinematics.
By doing so, only the forward kinematics of the open loops have to be known. The forward kinematics are straightforward and can be gained in an analytical way following the standard rules for serial kinematics (e.g. following the Denavit-Hartenberg convention). For

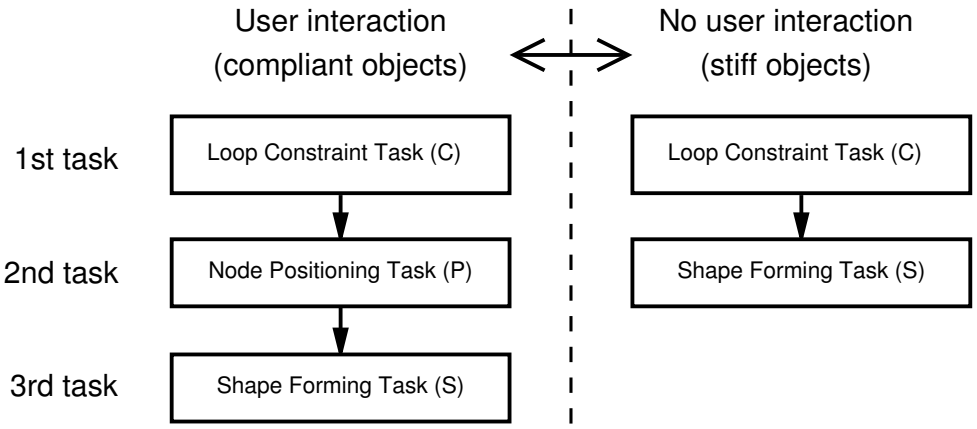


Figure 3.3: Hierarchy with and without user interaction

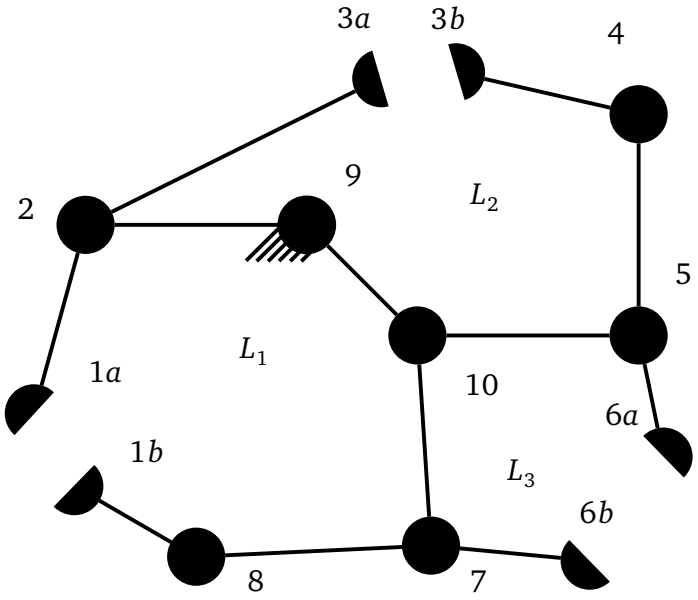


Figure 3.4: Mechanism with open loops

open loop j the forward kinematics can be written as follows:

$$\boldsymbol{x}_{Cja} = \boldsymbol{f}_{Cja}(\boldsymbol{\varphi}) \tag{3.9}$$
$$\boldsymbol{x}_{Cjb} = \boldsymbol{f}_{Cjb}(\boldsymbol{\varphi}) \tag{3.10}$$

Note that $\boldsymbol{x}_{Cja}$ describes the Cartesian position and orientation of one side of the cut and $\boldsymbol{x}_{Cjb}$ the Cartesian position and orientation of the other side. When the closed loop constraint is satisfied, the following equation must hold:

$$\boldsymbol{f}_{Cj} = \boldsymbol{x}_{Cja} - \boldsymbol{x}_{Cjb} = \boldsymbol{x}_{Cj} \stackrel{!}{=} \boldsymbol{0} \quad \text{with } j = 1 \dots l \tag{3.11}$$

Note that (3.11) must hold at any time due to imposed kinematic constraints of the kinematics. The Loop Constraint task is fulfilled in a differential way in the highest priority task of (3.5). Therefore task velocities $\dot{\boldsymbol{x}}_{Cj}$ must be calculated. These loop constraint velocities $\dot{\boldsymbol{x}}_{Cj}$ can be interpreted as relative velocities at the cuts (between a & b). The loop constraint error $\boldsymbol{e}_{Cj}$ is calculated from the difference of the desired (relative) position $\boldsymbol{x}_{Cjd} = \boldsymbol{0}$ and the actual (relative) position $\boldsymbol{x}_{Cj}$ which is calculated using (3.11). The task velocity $\dot{\boldsymbol{x}}_{Cj}$ is then calculated using (3.7). Beside task velocities also the Jacobian $\boldsymbol{J}_C$ of the task 'C' must be known in order to use (3.5). The time derivative of $\boldsymbol{f}_{Cj}$ leads to the Jacobian $\boldsymbol{J}_{Cj}$.

$$\dot{\boldsymbol{x}}_{Cj} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \boldsymbol{\omega} \end{bmatrix}_{Cj} = \nabla \boldsymbol{f}_{Cj}(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}} = \boldsymbol{J}_{Cj}(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}} = \begin{bmatrix} \boldsymbol{J}_p(\boldsymbol{\varphi}) \\ \boldsymbol{J}_o(\boldsymbol{\varphi}) \end{bmatrix}_{Cj} \dot{\boldsymbol{\varphi}}, \quad \text{with } j = 1 \dots l \tag{3.12}$$

The Jacobian $\boldsymbol{J}_{Cj}$ can be split into Jacobians $\boldsymbol{J}_{pCj}$ for positions and $\boldsymbol{J}_{oCj}$ for orientations. The desired velocity of the Loop Constraint task must be $\dot{\boldsymbol{x}}_{Cd} = \boldsymbol{0}$ at all times. If errors in the Loop Constraint task are made because of the inaccuracy of the weighted-damped Pseudoinverse or badly chosen joint angles in the first time step of the algorithm, this error will lead to a task velocity $\dot{\boldsymbol{x}}_C \neq \boldsymbol{0}$. This velocity originating from the task error $\boldsymbol{e}_C$ minimizes the task error so that the loop constraints are fulfilled. The velocities of all loops are given by $\dot{\boldsymbol{x}}_C = \begin{bmatrix} \dot{\boldsymbol{x}}_{C1}^T & \dots & \dot{\boldsymbol{x}}_{Cl}^T \end{bmatrix}^T$ with overall Jacobian $\boldsymbol{J}_C = \begin{bmatrix} \boldsymbol{J}_{C1}^T & \dots & \boldsymbol{J}_{Cl}^T \end{bmatrix}^T$.

3.2.5 Node Positioning Task

As mentioned before the Node Positioning task 'P' is executed on the Nullspace of the Loop Constraint task 'C'. Thus, when assuming that the Loop Constraint task is fulfilled, for the Node Positioning task we can virtually cut the loops and continue working with the resulting serial kinematics, which simplifies the inverse kinematics problem significantly. It is not important which path on the kinematic chain is taken, but, in order to reduce complexity, short paths (few joints) should be favored. In other words, different virtual cuts can be made for the Loop Constraint task and the Node Positioning task. The forward kinematics of one specific node in the kinematics follows from a composition of the transformations between the single nodes in the serial kinematic chain. The differential forward kinematics of node i is given by (3.13) and is again gained by differentiating the forward kinematics of a specific

node:

$$\dot{\boldsymbol{x}}_{Pi} = \left[\begin{array}{c} \dot{\boldsymbol{p}} \\ \boldsymbol{\omega} \end{array} \right]_{Pi} = \boldsymbol{J}_{Pi}(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}} = \left[\begin{array}{c} \boldsymbol{J}_p(\boldsymbol{\varphi}) \\ \boldsymbol{J}_o(\boldsymbol{\varphi}) \end{array} \right]_{Pi} \dot{\boldsymbol{\varphi}}, \quad \text{with } i = 1 \dots k \tag{3.13}$$

where $\boldsymbol{J}_{Pi}$ is the Jacobian of node i, which can be split into $\boldsymbol{J}_{pPi}(\boldsymbol{\varphi})$ for translational velocities and $\boldsymbol{J}_{oPi}(\boldsymbol{\varphi})$ for angular velocities. The translational velocity of node i is described by $\dot{\boldsymbol{p}} = \left[\begin{array}{ccc} \dot{p}_x & \dot{p}_y & \dot{p}_z \end{array} \right]^T$ and the angular velocity by $\boldsymbol{\omega} = \left[\begin{array}{ccc} \omega_x & \omega_y & \omega_z \end{array} \right]^T$. The overall Jacobian for the Node Positioning task 'P' is given by $\boldsymbol{J}_P = \left[\begin{array}{ccc} \boldsymbol{J}_{P1}^T & \dots & \boldsymbol{J}_{Pm}^T \end{array} \right]^T$. Note that, if only a few nodes are to be directly controlled, $\boldsymbol{J}_P$ is composed of the Jacobians of these nodes only. The Node Positioning task 'P' is used when haptic interaction takes place at specific nodes to avoid unrealistic haptic impressions.

Position- and Orientation-Errors of Nodes

The errors for the Node Positioning task must be derived in order to obtain the desired speed. The position error $\boldsymbol{e}_{pPi}$ for node i becomes:

$$\boldsymbol{e}_{pPi} = \boldsymbol{p}_{id} - \boldsymbol{p}_i \tag{3.14}$$

In (3.14) the position error is calculated as the difference of the desired (Cartesian) position $\boldsymbol{p}_{id}$ of node i and its actual position $\boldsymbol{p}_i$. In order to render the virtual shape one has to ensure that $\boldsymbol{p}_{id}$ lies on the virtual surface. For the orientation error $\boldsymbol{e}_{oPi}$ of node i one has to consider the actual rotation matrix $\boldsymbol{R}_i = \left[\begin{array}{ccc} \boldsymbol{n}_i & \boldsymbol{s}_i & \boldsymbol{a}_i \end{array} \right]$ and the desired rotation matrix $\boldsymbol{R}_{id} = \left[\begin{array}{ccc} \boldsymbol{n}_{id} & \boldsymbol{s}_{id} & \boldsymbol{a}_{id} \end{array} \right]$ of node i. The orientation error is calculated as described in [50]:

$$\boldsymbol{e}_{oPi} = \frac{1}{2}(\boldsymbol{n}_i \times \boldsymbol{n}_{id} + \boldsymbol{s}_i \times \boldsymbol{s}_{id} + \boldsymbol{a}_i \times \boldsymbol{a}_{id}) \tag{3.15}$$

3.2.6 Shape Forming Task

To position a node with a specific orientation in 3D space using the Node Positioning task, 6 DoF per node are needed. This means that for exact positioning of m nodes $6m$ DoF are necessary for the whole device. In most cases, such a high amount of DoF is neither available nor necessary for shape rendering. To display a specific shape the nodes do not have to be forced to a specific position, but the nodes can rather be positioned anywhere on the surface with an orientation in normal direction to the surface. To achieve this, only 3 DoF per node or $3m$ DoF for the whole device are needed. We developed a method where shape rendering can be derived from the Node Positioning task as explained in Section 3.2.5.

Distance to the Surface

When considering the positions of the nodes, the Euclidean distance to the surface d_i of each node i is of interest and should be controlled to a specific desired value d_{id} ($d_{id} = 0$ in most cases). Thus, the velocity of the nodes $\dot{\boldsymbol{p}}_i$ must be reformulated using the time derivative of the Euclidean distance $\dot{d}_i$. The temporal change of the distance $\dot{d}_i$ can be calculated by

projecting the velocity vector $\dot{\boldsymbol{p}}_i$ onto the normal vector $\boldsymbol{n}_i$ of the virtual surface at the actual node position (cf. Figure 3.5) by using the scalar product

$$\dot{d}_i = \boldsymbol{n}_i^T \cdot \dot{\boldsymbol{p}}_i. \tag{3.16}$$

Combining (3.16) with the translational part of (3.13) leads to

$$\dot{d}_i = \boldsymbol{n}_i^T \boldsymbol{J}_{pPi}(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}} = \boldsymbol{J}_{pSi}(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}}, \quad \text{with } \boldsymbol{J}_{pSi} = \boldsymbol{n}_i^T \boldsymbol{J}_{pPi}, \tag{3.17}$$

$$\boldsymbol{J}_{pS} = \begin{bmatrix} \boldsymbol{J}_{pS1}^T \ldots \boldsymbol{J}_{pSm}^T \end{bmatrix}^T. \tag{3.18}$$

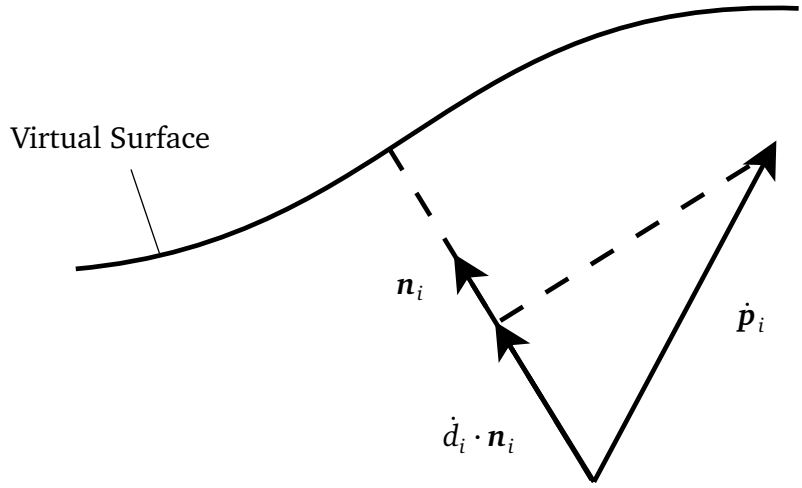


Figure 3.5: Projection of the node velocity in normal direction

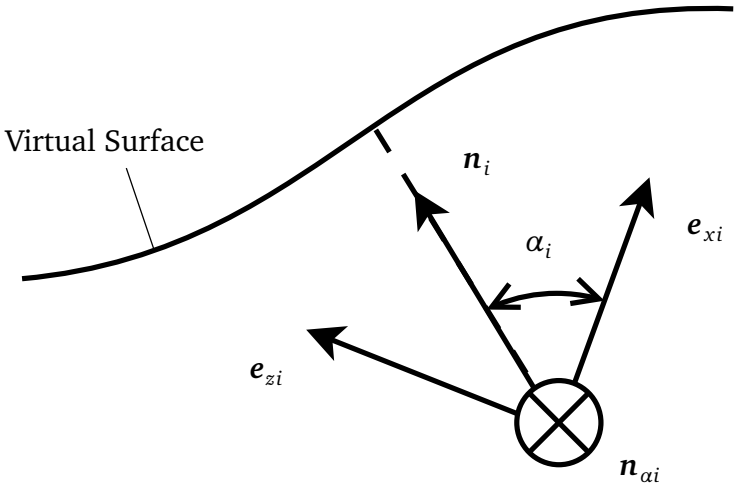


Figure 3.6: Area defining the direction of angular velocity

Angle between Node and Surface

In order to orient the nodes in an appropriate way, the body fixed z-axis $\boldsymbol{e}_{zi}$ and the normal vector of the surface $\boldsymbol{n}_i$ at point $\boldsymbol{p}_i$ must be aligned. This means that each node is allowed

to rotate about $\boldsymbol{n}_i$. Thus, 2 DoF per node are needed to control the orientation of the nodes. We define the angle between the body fixed x-axis $\boldsymbol{e}_{xi}$ and $\boldsymbol{n}_i$ and the angle between the body fixed y-axis $\boldsymbol{e}_{yi}$ and $\boldsymbol{n}_i$ to be α_i and β_i. In order to align $\boldsymbol{e}_{zi}$ with $\boldsymbol{n}_i$, α_i and β_i must be controlled to meet the desired values $\alpha_{id} = \beta_{id} = \pi/2$. Note that the desired values α_{id} and β_{id} might also be allowed to vary from $\pi/2$, if it is demanded by the application. The angular velocity $\boldsymbol{\omega}_i$ must be projected onto the rotation axes of α_i and β_i, respectively. These directions are $\boldsymbol{n}_{\alpha i}$ and $\boldsymbol{n}_{\beta i}$, the normal vectors of the planes which are spanned by $\boldsymbol{n}_i$ and $\boldsymbol{e}_{xi}$ or $\boldsymbol{e}_{yi}$, respectively. Figure 3.6 shows the plane with normal vector $\boldsymbol{n}_{\alpha i}$, which defines the rotation axis of α_i. The direction $\boldsymbol{n}_{\beta i}$ is defined analogously. The vector $\boldsymbol{n}_{\alpha i}$ ($\boldsymbol{n}_{\beta i}$) is calculated using the cross product of $\boldsymbol{n}_i$ and $\boldsymbol{e}_{xi}$ ($\boldsymbol{e}_{yi}$).

$$\boldsymbol{n}_{\alpha i} = \frac{\boldsymbol{n}_i \times \boldsymbol{e}_{xi}}{|\boldsymbol{n}_i \times \boldsymbol{e}_{xi}|}; \quad \boldsymbol{n}_{\beta i} = \frac{\boldsymbol{n}_i \times \boldsymbol{e}_{yi}}{|\boldsymbol{n}_i \times \boldsymbol{e}_{yi}|} \tag{3.19}$$

Note that the vectors $\boldsymbol{n}_i$, $\boldsymbol{e}_{xi}$ ($\boldsymbol{e}_{yi}$) and $\boldsymbol{n}_{\alpha i}$ ($\boldsymbol{n}_{\beta i}$) form a right-handed trihedron and if $\boldsymbol{n}_i$ and $\boldsymbol{e}_{zi}$ are aligned, $\alpha_i = \beta_i = \pi/2$. The angular velocity projected onto $\boldsymbol{n}_{\alpha i}$ ($\boldsymbol{n}_{\beta i}$) is given by

$$\omega_{\alpha i} = \boldsymbol{n}_{\alpha i}^T \cdot \boldsymbol{\omega}_i; \quad \omega_{\beta i} = \boldsymbol{n}_{\beta i}^T \cdot \boldsymbol{\omega}_i. \tag{3.20}$$

Combining (3.20) with the rotational part of (3.13) leads to

$$\omega_{\alpha i} = \boldsymbol{n}_{\alpha i}^T \boldsymbol{J}_{oPi}(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}} = \boldsymbol{J}_{\alpha Si}(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}}, \quad \text{with } \boldsymbol{J}_{\alpha Si} = \boldsymbol{n}_{\alpha i}^T \boldsymbol{J}_{oPi}, \tag{3.21}$$

$$\omega_{\beta i} = \boldsymbol{n}_{\beta i}^T \boldsymbol{J}_{oPi}(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}} = \boldsymbol{J}_{\beta Si}(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}}, \quad \text{with } \boldsymbol{J}_{\beta Si} = \boldsymbol{n}_{\beta i}^T \boldsymbol{J}_{oPi}, \tag{3.22}$$

$$\boldsymbol{J}_{oS} = \begin{bmatrix} \boldsymbol{J}_{\alpha S1} \ldots \boldsymbol{J}_{\alpha Sm} & \boldsymbol{J}_{\beta S1} \ldots \boldsymbol{J}_{\beta Sm} \end{bmatrix}^T. \tag{3.23}$$

Position- and Orientation-Errors of Nodes

The errors for the Shape Forming task must be derived in order to obtain the desired speed. The position error e_{pSi} for node i becomes:

$$e_{pSi} = d_{id} - d_i. \tag{3.24}$$

The orientation error e_{oSi} of node i is defined with the help of the angles α_i and β_i between the body-fixed vector $\boldsymbol{e}_{xi}$ ($\boldsymbol{e}_{yi}$) and the normal vector $\boldsymbol{n}_i$ at position $\boldsymbol{p}_i$:

$$\boldsymbol{e}_{oSi} = \begin{bmatrix} \alpha_{id} - \alpha_i & \beta_{id} - \beta_i \end{bmatrix}^T \tag{3.25}$$

$$\text{with } \alpha_i = \arccos\left(\frac{\boldsymbol{e}_{xi}^T \cdot \boldsymbol{n}_i}{|\boldsymbol{e}_{xi}^T \cdot \boldsymbol{n}_i|}\right) \tag{3.26}$$

$$\beta_i = \arccos\left(\frac{\boldsymbol{e}_{yi}^T \cdot \boldsymbol{n}_i}{|\boldsymbol{e}_{yi}^T \cdot \boldsymbol{n}_i|}\right) \tag{3.27}$$

Please note that α_i and β_i are well defined between 0 and π, which is sufficient for orienting the nodes.

3.2.7 Joint Limits & Singularities

The hierarchical Nullspace projection described in Section 3.2.1 makes use of the weighted-damped Pseudoinverse $\boldsymbol{J}^*$. The diagonal weighting matrix $\boldsymbol{W}(\boldsymbol{\varphi})$ is used for joint limit avoidance while the damping λ is used for robustness at singularities. If the joint φ_i is close to its joint limit, a high value appears at the corresponding position in $\boldsymbol{W}$ and consequently $\dot{\varphi}_i$ becomes small (0 at the joint limit). The damping matrix is calculated as described in [24]. A joint limit function $H(\boldsymbol{\varphi})$ is given by (3.28), where the joint limts for joint i are denoted by $\varphi_{i,max}$ and $\varphi_{i,min}$, respectively. The actual joint value is given by φ_i.

$$H(\boldsymbol{\varphi}) = \frac{1}{4}\sum_{i=1}^{n}\frac{(\varphi_{i,max}-\varphi_{i,min})^2}{(\varphi_{i,max}-\varphi_i)(\varphi_i-\varphi_{i,min})}. \tag{3.28}$$

The corresponding weighting factor w_i is calculated with the help of the gradient function $\frac{\partial H(\boldsymbol{\varphi})}{\partial \varphi_i}$ (see (3.29)(3.30)).

$$\frac{\partial H(\boldsymbol{\varphi})}{\partial \varphi_i} = \frac{(\varphi_{i,max}-\varphi_{i,min})^2(2\varphi_i-\varphi_{i,max}-\varphi_{i,min})}{4(\varphi_{i,max}-\varphi_i)^2(\varphi_i-\varphi_{i,min})^2} \tag{3.29}$$

$$w_i = \begin{cases} 1+\frac{\partial H}{\partial \varphi_i} & \text{if } \Delta|\frac{\partial H}{\partial \varphi_i}| \geq 0, \\ 1 & \text{if } \Delta|\frac{\partial H}{\partial \varphi_i}| < 0 \end{cases} \tag{3.30}$$

The weighting matrix $\boldsymbol{W}$ is built as a diagonal matrix with elements w_i:

$$\boldsymbol{W} = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \dots & 0 & w_n \end{bmatrix} \tag{3.31}$$

A very high damping factor λ will damp joint velocities $\dot{\boldsymbol{\varphi}}$ and thus, lead to a low Cartesian accuracy. On the other hand, low damping causes high joint velocities at singularities, which needs to be avoided.
If the shape task drives the kinematics close to a singular configuration and a low damping λ_S is used within this task, high joint velocities $\dot{\boldsymbol{\varphi}}_S$ of the shape task can affect the fulfillment of the constraint task even though the shape task is executed on the Nullspace $\boldsymbol{N}_C$ of the constraint task. This can be explained by an inaccurate calculation of the Nullspace $\boldsymbol{N}_C$: the weighted damped Pseudoinverse $\boldsymbol{J}_C^*$ (which is not the exact Inverse) is used for the calculation of $\boldsymbol{N}_C$. On the other hand, a high damping λ_S will avoid the fulfillment of the Shape Forming task and consequently leads to potentially large shape errors $\boldsymbol{e}_S$. In [41] we proposed the damping factor of the Shape Forming task to be dependent on the error resulting from the violation of constraints. We extend this approach by a dependency of λ_S on the two factors B_1 and B_2:

$$\lambda_S = \frac{\lambda_0}{B_1 B_2}. \tag{3.32}$$

The factors B_1 and B_2 can take values between zero and one. The smallest possible value for λ_S is λ_0. The blending factor B_1 is a function of the normalized constraint error e_{Cn}:

$$e_{Cn} = \max_j(|\boldsymbol{e}_{Cpj}| + p_o|\boldsymbol{e}_{Coj}|), \tag{3.33}$$

$$B_1 = \frac{e^{(-\alpha\frac{e_{Cn}}{d}-\delta)}}{1+e^{(-\alpha\frac{e_{Cn}}{d}-\delta)}}. \tag{3.34}$$

The normalized constraint error e_{Cn} is the maximum of the weighted sums of the position error $\boldsymbol{e}_{Cpj}$ and the orientation error $\boldsymbol{e}_{Coj}$ of all loops, with gain p_o. In (3.34) α, d and δ are used to change the characteristics of the blending function. For a detailed description of the blending function please refer to [24]. If, like in [41], only B_1 is used ($B_2 = const = 1$) an error in the constraint task has to occur first in order to gain a small blending factor B_1. Thus, in some cases the loop constraints can be badly violated. We therefore suggest to use the additional blending factor B_2, which is defined as $B_2 = \min(\boldsymbol{W}^{-1})$, the smallest element of the inverse weighting matrix $\boldsymbol{W}$. This means that B_2 is defined by the weighting factor of the joint which is closest to its joint limit. Note that B_2 only takes care of singularities caused by joint limits. Singularities in the workspace are still considered by B_1. As soon as the kinematics comes close to a singular configuration B_1 and/or B_2 decrease.

3.2.8 Shape Rendering

The method driving all nodes using the Shape Forming task and simultaneously taking loop constraints into account by means of the Loop Constraint task is called *Shape Rendering*. Shape rendering combines the two tasks into a hierarchical framework. To fulfill the shape task $3m$ DoF are needed. If that amount of DoF (or more) is available, the shape rendering interface will render the shape without errors, as long as no joint limits are reached. When using the Shape Forming task, nodes are not fixed to a specific position/orientation. The nodes are allowed to slide on the surface and rotate around their $\boldsymbol{e}_z$-axes. This behavior can be observed when compliant objects are rendered, because the shape changes due to haptic interaction. The changed shape depends on the deformation model used and will not be discussed further here. Some examples for deformation models can be found in [51], [52], [53], [54], [55], [56] or [57]. In the following, we distinguish between *Static* and *Dynamic* Shape Rendering.

Static Shape Rendering

We refer to Static Shape Rendering if an object which is to be rendered, keeps a constant shape or only changes the shape slowly. This means that in (3.7) $\dot{\boldsymbol{x}}_{kd}$ is small or zero for the Shape Forming task. This is for instance the case when initially adapting to an object with constant shape. The kinematics will in most cases be in a configuration which does not render the shape and the task error $\boldsymbol{e}_S \neq \boldsymbol{0}$, but the desired task velocity is $\dot{\boldsymbol{x}}_{Sd} = \boldsymbol{0}$. In this case the task velocity $\dot{\boldsymbol{x}}_S$ and consequently the shape adaption will be driven by the task error $\boldsymbol{e}_S$. For stability reasons the gain K_S must be set to a small value and thus, $\dot{\boldsymbol{x}}_S$ will also be small. This means that the shape will not be adapted immediately, but after some time. Once the static shape is rendered ($\boldsymbol{e}_S = \boldsymbol{0}$), a changing shape can be followed faster, because $\dot{\boldsymbol{x}}_{Sd} \neq \boldsymbol{0}$.

Dynamic Shape Rendering

We refer to *Dynamic Shape Rendering* if an object shape changes fast, e.g., in case of a sinusoidal movement. If the shape changes too fast, the shape rendering device will have problems to follow the desired shape, because the damping λ_S and the weighting-matrix $\boldsymbol{W}_S$ of the Pseudoinverse of the Jacobian $\boldsymbol{J}_S$ will limit the joint velocities $\dot{\boldsymbol{\varphi}}_S$ (cf. (3.4) and (3.5)).

3.2.9 Dynamic Shape Rendering with User Interaction

If a subject interacts with the shape rendering interface and a compliant object is to be rendered, the virtual shape and thus, the configuration of the interface will change. In this case, the interface must be equipped with sensors, e.g. pressure sensors, in order to detect the user-interaction. If all nodes are controlled in a shape forming task, the subject can feel shear forces because of the nodes moving parallel to the virtual surface. In order to avoid this, we suggest adding a third task to the hierarchical framework. The nodes where haptic interaction takes place are controlled to exact positions and orientations, in a Node Positioning task. At most $\frac{M}{6}$ nodes can be controlled to exact positions and orientations because in order to do so, 6 DoF per node are needed. In contrast, 3 Dof are needed to control one node in a shape forming task, where the node is positioned somewhere on the surface in normal direction to it. The exactly controlled nodes gain higher priority over the other nodes controlled in the Shape Forming task. If the kinematics is highly redundant, the shape will be approximated like when controlling all nodes in the Shape Forming task (if no joint limits are reached). But if not enough DoF are available, shape errors will occur.
The Jacobian for the Node Positioning task, which is executed in the Nullspace of the constraint task, is composed of the Jacobians $\boldsymbol{J}_{Pi}$ of the nodes which are controlled in the Node Positioning task. If for example node 1, 3 and 10 are to be controlled in the Node Positioning task $\boldsymbol{J}_P$ becomes:

$$\boldsymbol{J}_P = \begin{bmatrix} \boldsymbol{J}_{P1}^T & \boldsymbol{J}_{P3}^T & \boldsymbol{J}_{P10}^T \end{bmatrix}^T. \tag{3.35}$$

In order to guarantee the fulfillment of the Node Positioning task one has to guarantee: $0 \leq \sum r \leq \frac{M}{6}$, with r the number of nodes to be controlled in the Node Positioning task. This means that with this approach maximal $\frac{M}{6}$ nodes can be controlled. This only applies if the desired positions lie inside the workspace of the mechanism (no joint limits are reached). Further, please note that contacts occurring between the nodes are not discussed here. In such a case unrealistic haptic impressions can appear, when the area of the operator's fingertip (diameter approx. 1 cm) is smaller than the distance of the nodes. Then the operator feels the compliance of the elastic layer. Avoiding such unrealistic haptic impressions would require modelling the elastic layer and introducing a new task, which controls the point of interaction (between the nodes) to specific positions. To this end, one would need to model the elasticity of the layer as a function of tension in the layer-material. The problem arising from the elasticity of the layer will, however, be dissolved automatically once a shape rendering interface with a higher resolution (node distance smaller than the fingertip) will become available. In this case the fingertip of the operator will always touch at least one node. Finally, the rendering of compliant objects could be achieved by adding sensors to the elastic layer to determine the desired dynamic displacement of the nodes in response to the

selected deformation model. Alternatively and if only minor deformations are required one could improve the elastic layer to control the elasticity of the layer itself. This could e.g. be achieved by particle jamming as proposed in [58].

3.3 Shape Descriptions: Implicit Surfaces and Polygon Sets

The input of the Shape Forming task described in 3.2.6 uses task space velocities $\dot{\boldsymbol{x}}_S$ ($\dot{\boldsymbol{d}}$, $\boldsymbol{\omega}_\alpha$, $\boldsymbol{\omega}_\beta$). For the calculation of the task space velocities all actual node distances d_i and the corresponding normal vectors of the virtual surface $\boldsymbol{n}_i$ at position $\boldsymbol{p}_i$ must be known. Position $\boldsymbol{p}_i$ is calculated with the help of the forward kinematics of the cut mechanism. Note that all joint angles are assumed to be known. The position/orientation errors must be calculated from the actual node position and the virtual surface, which is to be rendered. In the following two possible descriptions of virtual surfaces are discussed: implicit surfaces and polygon sets. Implicit surfaces are easy to handle but complicated surfaces are difficult to describe. With polygon sets, complicated object shapes can be specified without effort. We show how the distances d_i and the normal vectors $\boldsymbol{n}_i$ required for the shape rendering can be obtained from these descriptions, so that implicit surfaces and surfaces defined by polygon sets can be handled.

Implicit Surface

Many object surfaces can be described by an implicit equation. An implicit surface in 3D Euclidean space is defined as a function $f(\boldsymbol{p}) = f(p_x, p_y, p_z) = \hat{d}$, where $\hat{d} = 0$ on the surface. The function value is $\hat{d} > 0$ outside the object and $\hat{d} < 0$ inside. In most cases the function value $\hat{d}$ does not match the Euclidean distance d to the surface. The Euclidean distance d, however, can be calculated with the help of the algorithm presented in [59], where the closest surface point $\boldsymbol{p}_S$ of a point $\boldsymbol{p}$ in space is calculated in a few iterations. Once the surface point $\boldsymbol{p}_S$ is known, the Euclidean distance can be calculated by:

$$d = \operatorname{sgn}(f(\boldsymbol{p}))|\boldsymbol{p_S} - \boldsymbol{p}| \tag{3.36}$$

The normal vector $\boldsymbol{n}$ at position $\begin{bmatrix} x & y & z \end{bmatrix}^T$ can be calculated with the gradient $\nabla f(x,y,z)$ of the implicit function:

$$\boldsymbol{n}(x,y,z) = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T = \frac{\nabla f(x,y,z)}{|\nabla f(x,y,z)|} \tag{3.37}$$

Polygon Set

In many cases, implicit surface descriptions are not available, especially when complex objects are considered. Inspired by [60] and [61] we developed a method where node distance and normal vectors are gained from polygon sets (triangle meshes). In [60] active constraints (restricting planes) were used to define Lagrange multipliers to be solved by means of optimization. The optimization delivers the position of the so-called *god-object*, which is the closest surface point to the haptic interaction point that penetrates the virtual object. In contrast we do not base our algorithm on optimization, but we rather aim for a method which

uses offline calculations to generate a database which delivers information about neighboring primitives of the polygon data set by orthogonally projecting $\boldsymbol{p}_i$ onto the surfaces/edges and vertices of interest. No distinction of convex and concave constraints (for convex and concave surface regions) is needed because this distinction is indirectly performed by our algorithm. By doing so, at each time step only a small part of the dataset has to be taken into account, which accelerates the execution of the algorithm. In [61] a similar approach was presented, but the dataset was built based on 3 primitives (Vertex, Line, Polygon) and in each iteration it was checked if the surface point is on the primitive or on its neighboring primitives. This means that if for example in iteration j the surface point $\boldsymbol{p}_S$ lies on a polygon, in the next iteration $(j+1)$ it is checked if $\boldsymbol{p}_S$ lies on that polygon or on its surrounding primitives (edges and vertices). Our algorithm also takes into account all surrounding polygons and thus, achieves a more accurate solution at iteration $j+1$ compared to [61]. In contrast the algorithm in [61] would find a surface point $\boldsymbol{p}_S$ on a neighboring polygon in iteration $j+2$.
We assume that the polygon set is available in STL-format, where vertices and normal vectors are given for each polygon. This dataset is preprocessed offline in a first step to make it usable for the online phase of our algorithm.

Offline Phase: In the Offline Phase, the given information from the polygon set is converted into an extended data set. We add the following information to the data set:

- Polygon ID
- Neighbor polygons of vertex 1, 2 and 3 of a certain polygon
- Neighbor polygons of edges 1, 2 and 3 of a certain polygon

Table 3.1 shows the elements of the STL-data set and the extended data set. The last column shows values for the example polygon set shown in Figure 3.7.

Tabular 3.1: STL-data set and its extension

STL-Dataset	Vertex 1	Cartesian position
	Vertex 2	Cartesian position
	Vertex 3	Cartesian position
	Normal Vector	Cartesian vector
Extension	Polygon ID	$P7$
	Neighbor-IDs Vertex 1	$P1$,$P2$,$P3$,$P8$,$P9$
	Neighbor-IDs Vertex 2	$P3$,$P4$,$P5$,$P6$
	Neighbor-IDs Vertex 3	$P6$,$P9$,$P10$,$P11$
	Neighbor-IDs Edge 1	$P3$
	Neighbor-IDs Edge 2	$P6$
	Neighbor-IDs Edge 3	$P9$

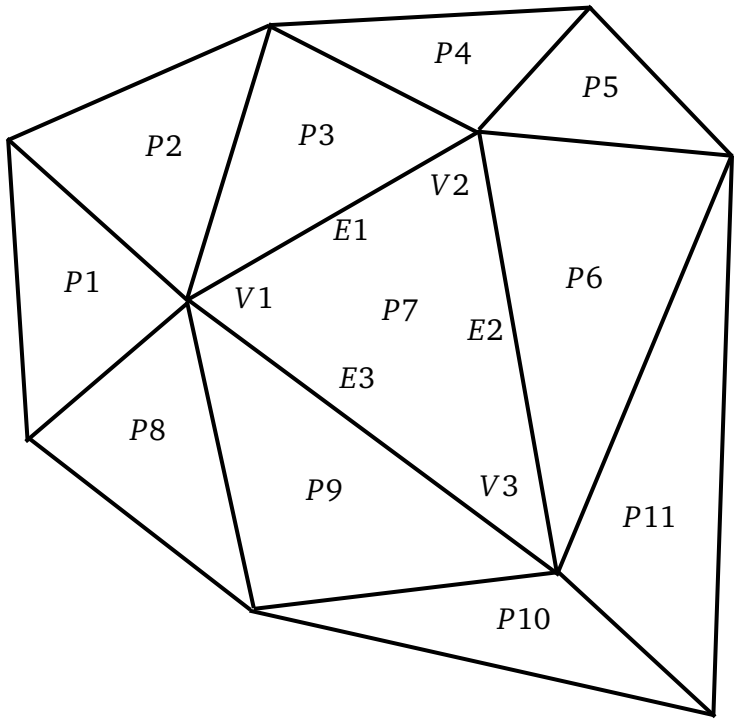


Figure 3.7: Example of polygons with IDs

Online Phase: In the Online Phase the previously generated data set is used to find the closest surface point $\boldsymbol{p}_{Si}$ and the corresponding normal vector $\boldsymbol{n}_i$. The distance d_i is defined by the Euclidean distance between $\boldsymbol{p}_{Si}$ and $\boldsymbol{p}_i$. At any time step the distances d_i of all nodes and normal vectors $\boldsymbol{n}_i$ at positions $\boldsymbol{p}_i$ are needed to calculate the actual task-errors (cf. (3.24), (3.25), (3.26), (3.27)) and to gain the Jacobians $\boldsymbol{J}_{Si}(\boldsymbol{\varphi})$ for the shape task ((3.17), (3.23)). The most obvious solution to get the distances d_i would be to identify the closest vertex of the polygon set to the node of interest to define $\boldsymbol{p}_{Si}$, whereas $\boldsymbol{p}_{Si}$ is a 3D Cartesian position on the virtual surface. This approach can lead to a surface point $\boldsymbol{p}_S$ which has a high distance to $\boldsymbol{p}_i$ even if $\boldsymbol{p}_i$ is close to the virtual surface. This especially can be observed, when large polygons are enclosed in the dataset and $\boldsymbol{p}_i$ is close to such a large polygon, but far from the vertices defining the polygon. A better solution is consequently to obtain the distances of $\boldsymbol{p}_i$ to the closest polygon face and pick this distance as the demanded distance d_i. This, however, may lead to situations, in which the orthogonal projected point $\boldsymbol{p}_{Si}$ lies outside of the corresponding polygon, which means that a projection to an edge or a vertex would produce a more accurate solution for the distance (cf. Figure 3.9 and 3.10). We therefore suggest an algorithm, which takes into account projections to faces and edges, for finding the minimal distance and normal vector. A flowchart of the algorithm is shown in Figure 3.8.

The algorithm takes advantage of the dataset created in the Offline Phase. In the first step, the closest vertex is determined from either all polygons or a meaningful choice of polygons (e.g. the polygons close to the node of interest). The distance is determined by the Euclidean distance of $\boldsymbol{p}_i$ to the vertex. The offline dataset provides the polygons, which share this closest vertex (involved polygons). The point $\boldsymbol{p}_i$ is then orthogonally projected to all surfaces of the involved polygons and it is checked how many projected points lie inside the corresponding polygons. If multiple projected points lie inside the polygons, the one

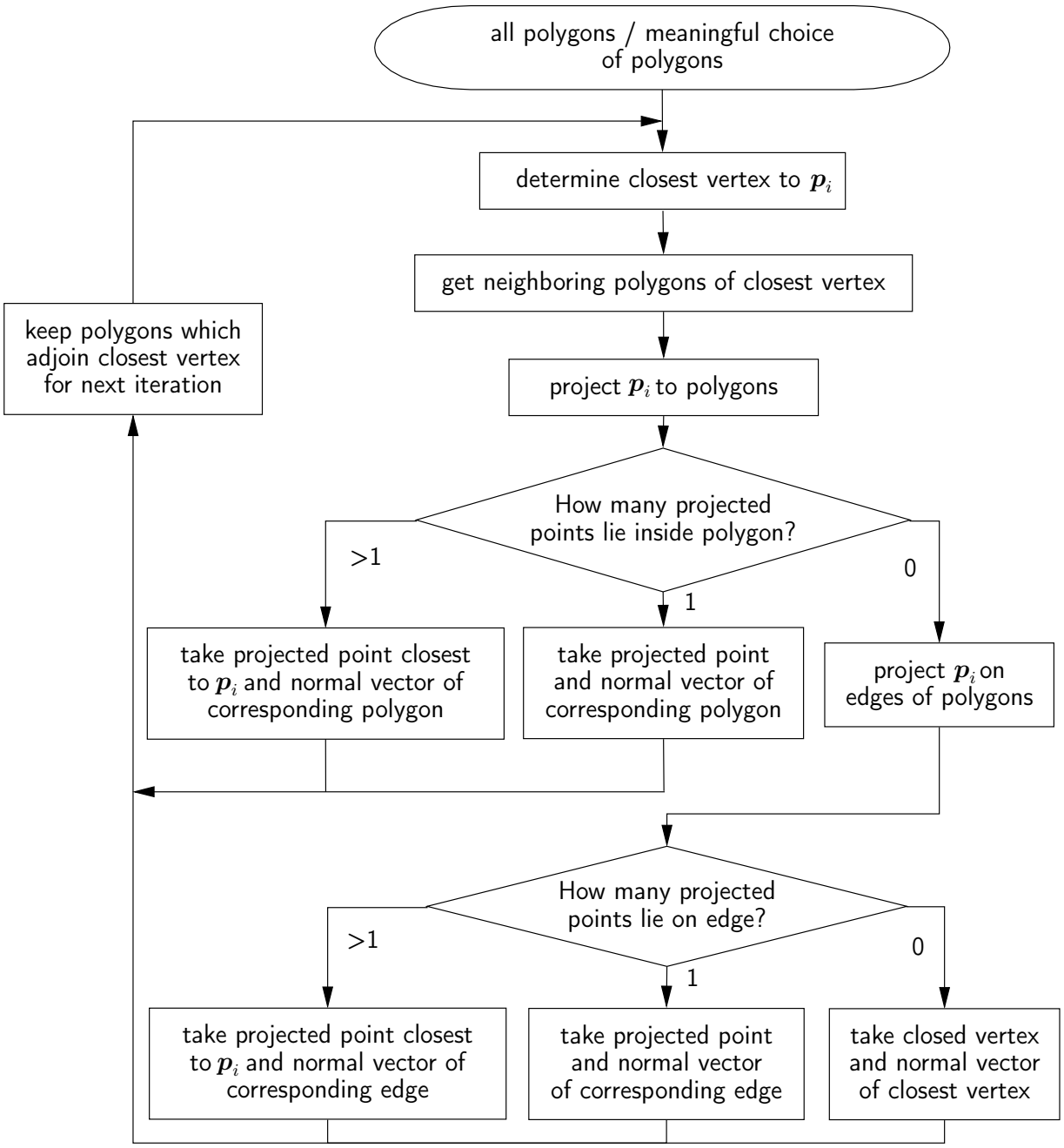
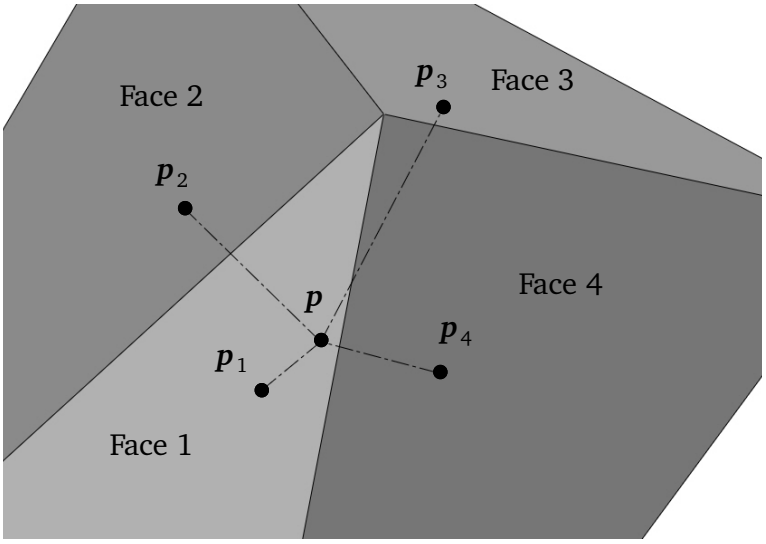
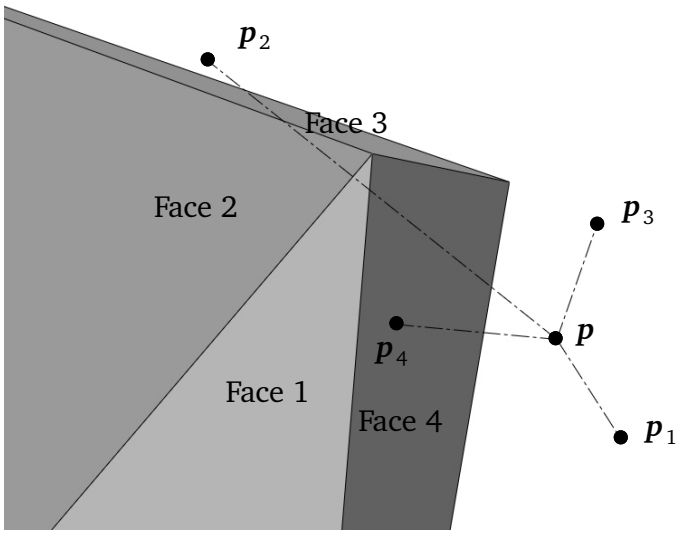


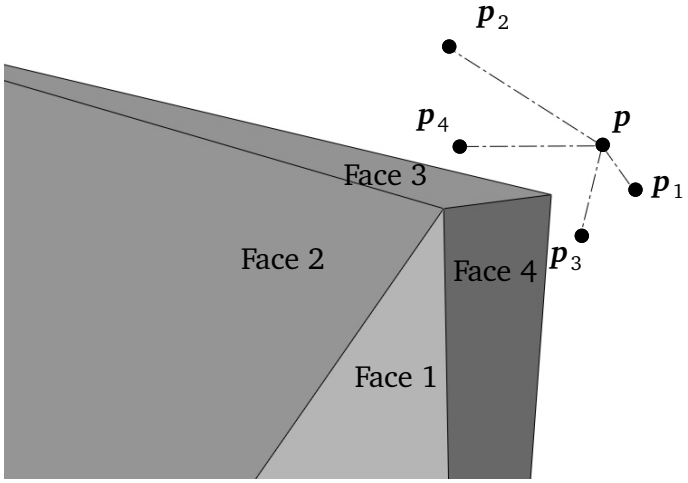
Figure 3.8: Algorithm to find distance d and normal vector $\boldsymbol{n}$ from a polygon set



(a) Multiple projected points ($p_1, \ldots, p_4$) are inside polygon

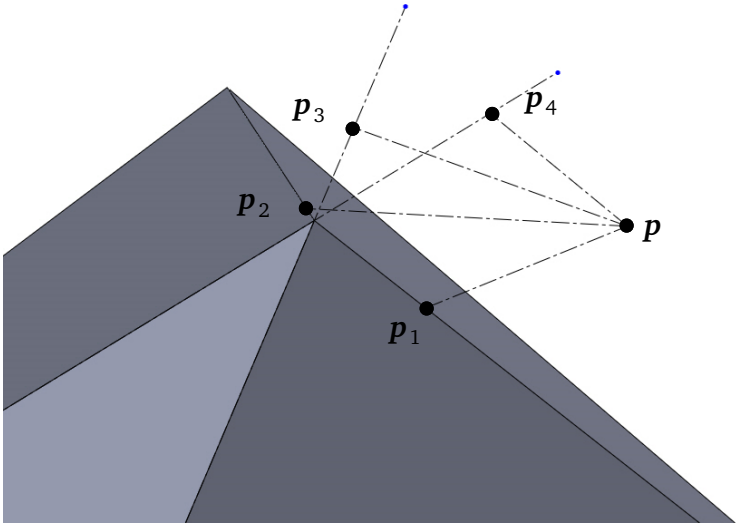


(b) One projected point (p_4) is inside polygon

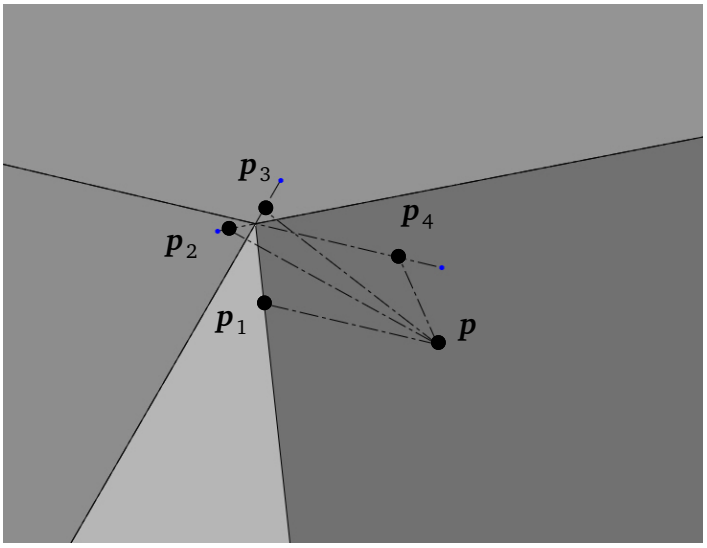


(c) No projected point is inside polygon

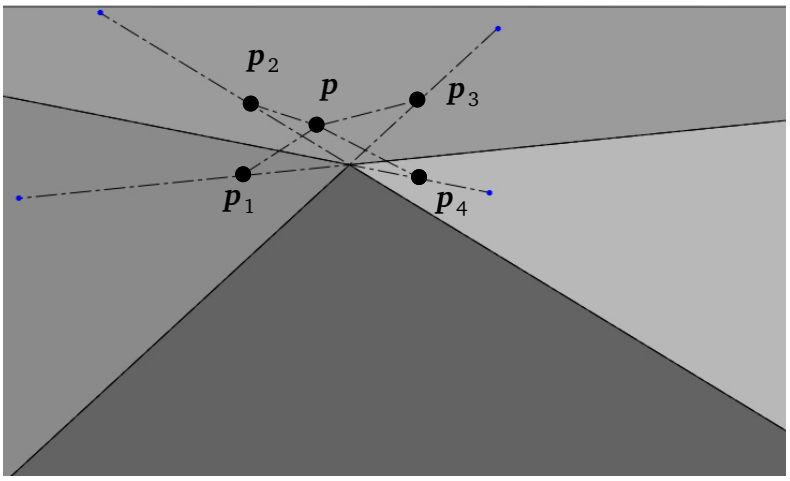
Figure 3.9: Projection on surfaces



(a) Multiple projected points (p_1, p_2) are on edge



(b) One projected point (p_1) is on an edge



(c) No projected point is on edge

Figure 3.10: Projection on edges

with the smallest distance to $\boldsymbol{p}_i$ is chosen to be $\boldsymbol{p}_{Si}$. If only one projected point lies inside the corresponding polygon this point is set to be $\boldsymbol{p}_{Si}$ and if none of the projected points lies inside, further projections to edges must be made: The point $\boldsymbol{p}_i$ is orthogonally projected to all edges, which share the previously determined vertex (closest vertex). Analog to the plane projections, it is checked how many projected points lie on the corresponding edge. If more than one projected point is on an edge, the one with the shortest distance is chosen to be $\boldsymbol{p}_{Si}$. If only one point lies on the corresponding edge, this point is chosen. If no projected point is on the edge, the closest vertex determined at the beginning is defined to be $\boldsymbol{p}_{Si}$.
To summarize we have three cases where $\boldsymbol{p}_{Si}$ can be located: on a polygon, edge or vertex. Depending on these cases, the corresponding normal vectors are calculated: if the point $\boldsymbol{p}_{Si}$ lies on a polygon, the normal vector $\boldsymbol{n}_i$ is given by the normal vector of the polygon $\boldsymbol{n}_p$. If the point $\boldsymbol{p}_{Si}$ lies on an edge $\boldsymbol{n}_i = \frac{\boldsymbol{n}_{p1}+\boldsymbol{n}_{p2}}{|\boldsymbol{n}_{p1}+\boldsymbol{n}_{p2}|}$, with $\boldsymbol{n}_{p1}$ and $\boldsymbol{n}_{p2}$ the normal vectors of the planes adjoining the edge. The normal vector at a vertex is calculated by the normalized mean of the normal vectors of all r adjoining planes: $n_i = \frac{\boldsymbol{n}_{p1}+\ldots+\boldsymbol{n}_{pr}}{|\boldsymbol{n}_{p1}+\ldots+\boldsymbol{n}_{pr}|}$. The involved polygons (polygons adjoining the closest vertex) are used as input for the next iteration. In Figure 3.9 and 3.10 the different cases considered in the algorithm are visualized.

3.4 Results

In this section, the previously explained methods for controlling SRIs are applied to a kinematics with 24 DoF, the Formable Object (FO) presented in chapter 2. The Formable Object consists of 9 nodes, which are connected by links with 4 DoF each. The nodes are connected to form a network of nodes with 4 closed kinematic chains. The FO is to be positioned in space by a robotic device, which is not considered here. Node 1 (fixed node) is thereby positioned in the center of the region which is to be rendered by the FO. Nodes 2-9 (free nodes) are controlled using the methods described in this chapter. As the FO is to be positioned in space by a robotic device and rotations about the $\boldsymbol{e}_z$-axis of node 1 are allowed, we add this rotational DoF to the kinematics of the device. Thus, we are dealing with 25 DoF whereas the additional/redundant DoF helps in finding a better configuration to render a shape in some cases.

3.4.1 Shape Rendering

This section demonstrates the shape rendering of the FO. As shape input, a 3D elliptic surface is used, which is described implicitly:

$$f(x,y,z) = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = \hat{d}, \tag{3.38}$$

$$\boldsymbol{n}(x,y,z) = \begin{bmatrix} \frac{2x}{a^2} & \frac{2y}{b^2} & \frac{2z}{c^2} \end{bmatrix}^T. \tag{3.39}$$

We chose the elliptic shape as test shape as any complex shape can be assembled of concave and convex elliptic shapes. The simulation was performed with a sampling rate of 1 kHz. A wide range of performance measures for haptic interfaces can be found in [23]. As these measures are specified for classical haptic interfaces with one interaction point, we introduce

the Shape Forming Index (SFI). In the following sections, the shape rendering performance is measured and compared with the help of the SFI, which is defined as the mean of the normalized position and orientation errors of all nodes:

$$SFI = \frac{\arctan\left(K_p\overline{\left[\begin{array}{ccc} |e_{p1}| & \dots & |e_{pm}| \end{array}\right]}\right) + \overline{\left[\begin{array}{ccc} |e_{o1}| & \dots & |e_{om}| \end{array}\right]}}{\pi}, \tag{3.40}$$

with e_{pi} and e_{oi} the position and orientation errors of all nodes $1 \dots m$. The SFI can take values between zero and one. The gain K_p is used to generate a balanced weighting between position errors $\boldsymbol{e}_p$ and orientation errors $\boldsymbol{e}_o$. In order to equally weight a position error of $\boldsymbol{e}_p = 0.001$ m and an orientation error of $\boldsymbol{e}_o = 1° = 0.0175$ rad we chose a gain of $K_p = 17.5$.

Static Shape Rendering

At the beginning of the simulation, we assume the kinematics in a flat configuration (cf. Figure 2.23a). Node 1 is placed at the center of the surface to be approximated by the FO. As mentioned before node 1 is allowed to rotate around its $\boldsymbol{e}_z$-axis, which is useful to find a better orientation of the FO to render the desired shape. We simulated the adaptation of the FO to a spherical shape ($a = b = c = R$) where R is the radius of the sphere. In Figure 3.11 one can see how the blending factors B_1 and B_2 behave and the SFI visualizes how well the shape is approximated:
The radius R is slowly changed from 0.2 m to 0.1 m and backwards as well as from 0.2 m to 0.05 m. In the very beginning one can see that the blending factor B_1 rises from 0 to 1. This means that loop constraints are slightly violated at the very beginning. Because in every iteration of the algorithm the joint angles $\boldsymbol{\varphi}$ are calculated by integrating the joint velocities $\dot{\boldsymbol{\varphi}}$, initial joint angles are needed for the integrator. These angles are numerically not accurate enough, such that the loop constraints are slightly violated, and thus B_1 is found to be close to zero. The error in the Loop Constraint task is eliminated by the control algorithm after 70 ms. Note that this small error in the Loop Constraint task does not affect the hardware, which is controlled by the algorithm. Such small errors in the loop constraint are compensated by backlash in the mechanism and on the other side cannot be measured because the sensor accuracy is not high enough. As soon as the Loop Constraint task is fulfilled the SFI decreases to 0 which means that the FO adapts the desired shape ($R = 0.2$m). From $t = 1$ s$\dots$4 s the shape changes from $R = 0.2$ m to $R = 0.1$ m and backwards. Slight shape errors are made while the radius changes, because the task velocity of the shape task $\dot{\boldsymbol{x}}_S$ and the related joint velocities $\dot{\boldsymbol{\varphi}}_S$ are reduced because of the weighted damped Pseudoinverse (cf. (3.4)). As soon as the radius remains constant, the shape error is eliminated by the error term in (3.7). From $t = 5$ s$\dots$8 s the radius changes from 0.2 m to 0.05 m and backwards. A sphere with radius 0.05 m cannot be rendered by the FO without shape error, because joint limits are reached. One can see that the blending factor B_2 decreases to zero in the very beginning. This means that a very high damping is used for the shape task and thus, joint velocities $\dot{\boldsymbol{\varphi}}_S$ are inhibited. The blending factor B_2 depends on the weighting matrix $\boldsymbol{W}$. If B_2 is not low enough to guarantee the fulfillment of the Loop Constraint task, B_1 helps to further increase the damping λ_S. This behavior can be observed after 6 s. Figures 3.12 and 3.13 show the simulation of the FO adapting to the desired shapes at time $t = 2.5$ s ($R = 0.1$ m) and

$t = 6.5$ s ($R = 0.05$ m), respectively. One can see that a radius of $R = 0.1$ m is rendered without shape errors, but rendering a sphere with $R = 0.05$ m is not possible without errors.

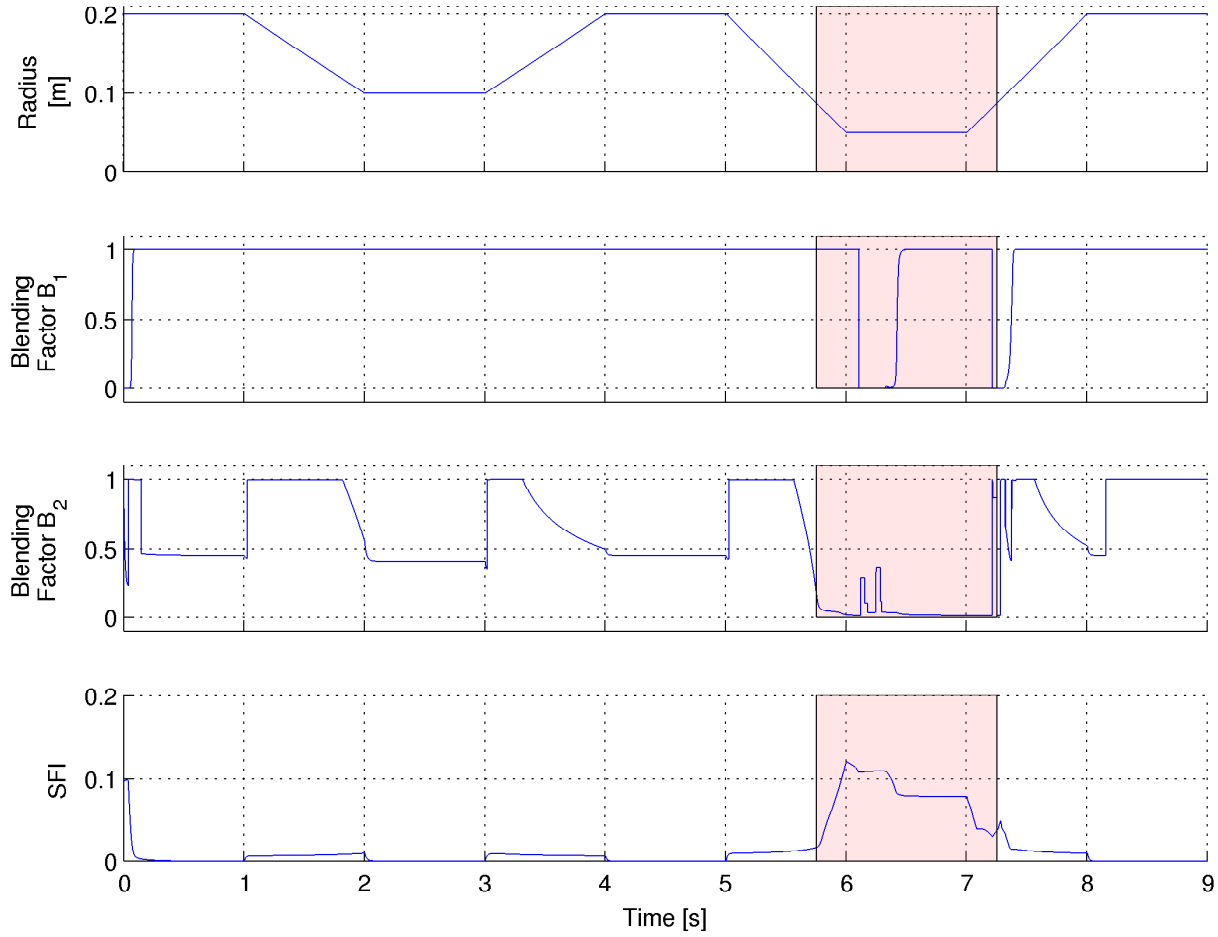


Figure 3.11: Static shape rendering

Figures 3.14 - 3.19 show simulations of the FO rendering several shapes. Cylindrical, spherical and elliptical shapes are visualized. It is shown how convex and concave shapes can be rendered.

Dynamic Shape Rendering

In order to evaluate the performance of the algorithm to render varying shapes, simulations with dynamically changing shapes were performed. The changing radius R, blending factors B_1 and B_2 and the SFI are shown in Figure 3.20. The radius R was changed with different frequencies, namely 1 Hz, 2 Hz and 3 Hz. These frequencies were chosen because subjects select frequencies of about 2 Hz when being asked to perform cyclical flexion-extension movements with their fingers [62] and thus, we expect shape changes in this frequency range when haptic interaction takes place. One can see that higher frequencies produce larger shape errors. The faster the shape/radius changes the larger values can be observed for the SFI. The

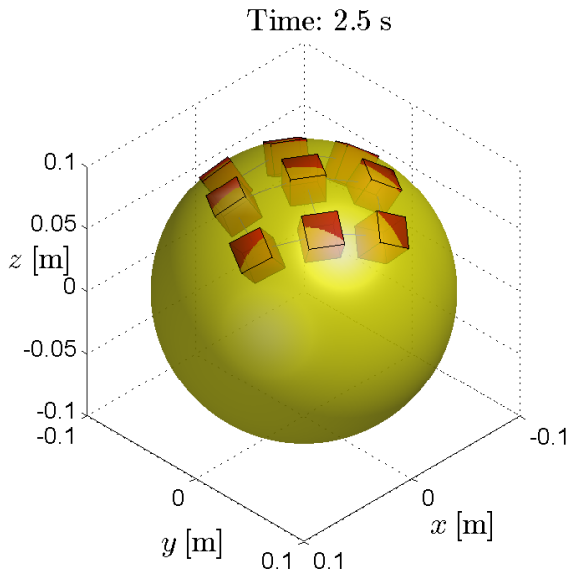


Figure 3.12: FO rendering a sphere with radius $R = 0.1$m

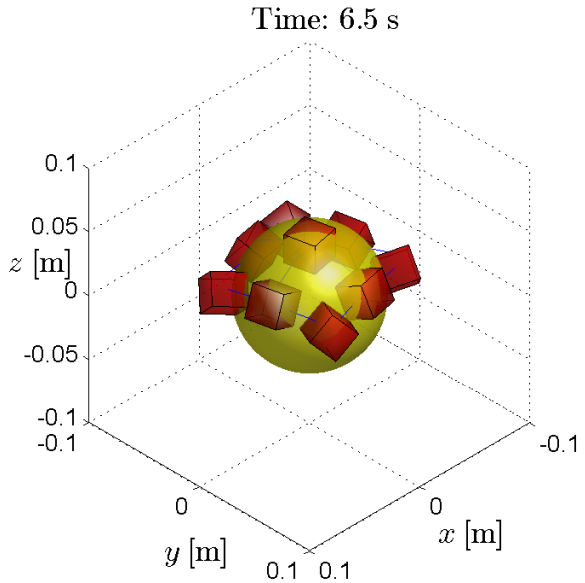


Figure 3.13: FO rendering a sphere with radius $R = 0.05$m

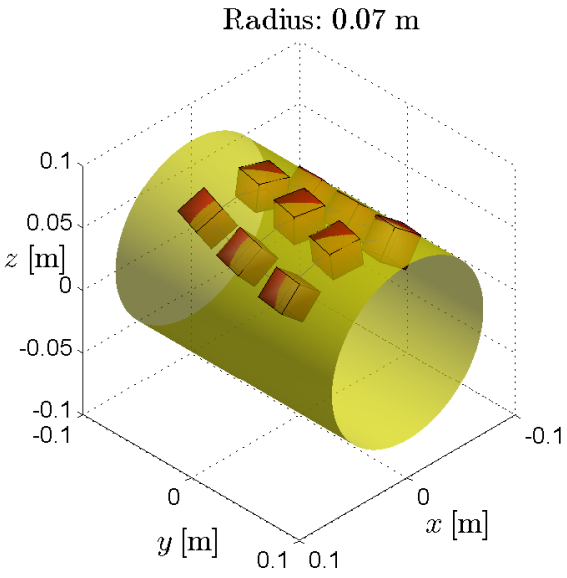


Figure 3.14: Convex rendering of cylindrical shape with $R = 0.07$ m

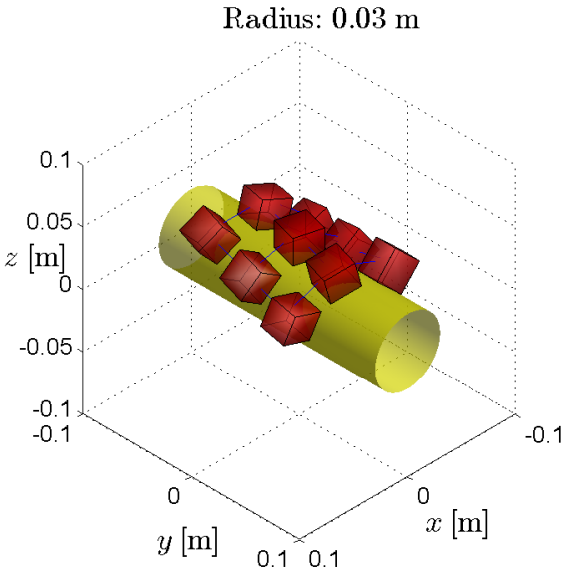


Figure 3.15: Concave rendering of cylindrical shape with $R = 0.03$ m

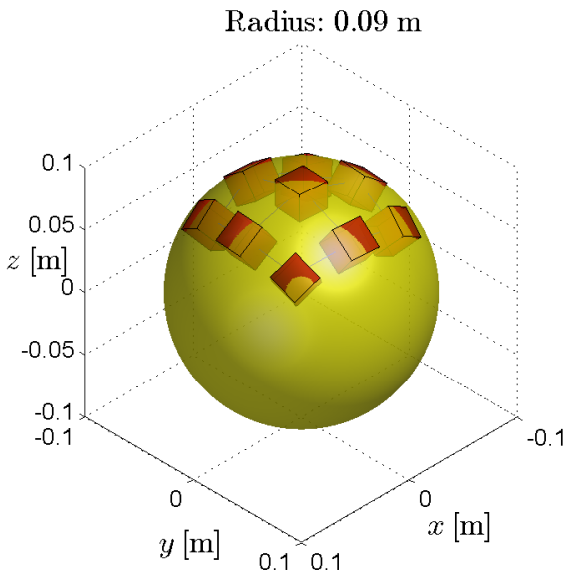


Figure 3.16: Convex rendering of spherical shape with $R = 0.09$ m

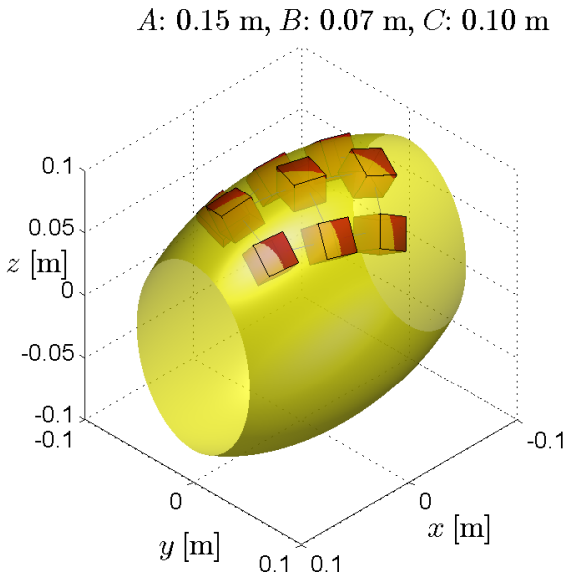


Figure 3.18: Convex rendering of elliptical shape with $A = 0.15$ m, $B = 0.07$ m and $C = 0.10$ m

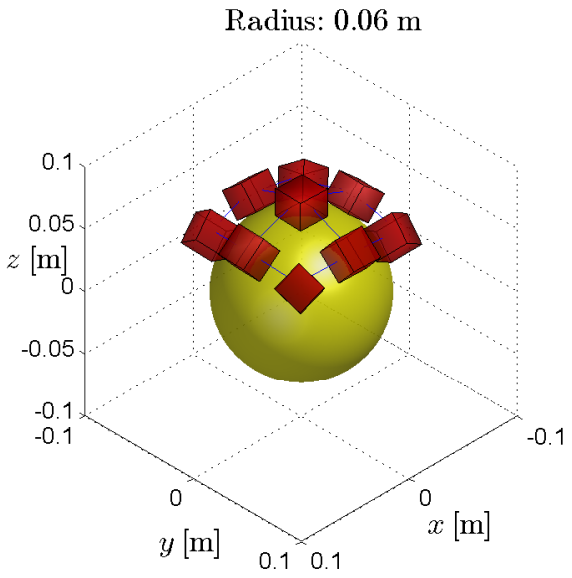


Figure 3.17: Concave rendering of spherical shape with $R = 0.06$ m

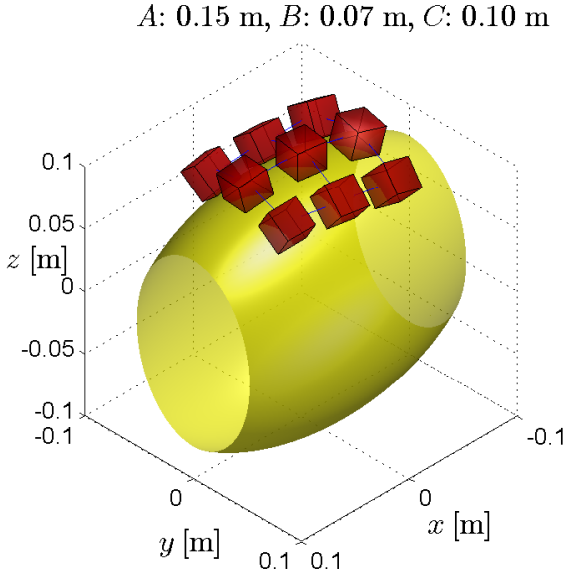


Figure 3.19: Concave rendering of elliptical shape with $A = 0.15$ m, $B = 0.07$ m and $C = 0.10$ m

reason is the same as described in the previous section: joint velocities $\dot{\varphi}_S$ are damped by the damping factor λ_S and the weighting matrix $\boldsymbol{W}$. The errors, which are made, must be eliminated by the error term in (3.7), which works the better the slower the desired shape changes.

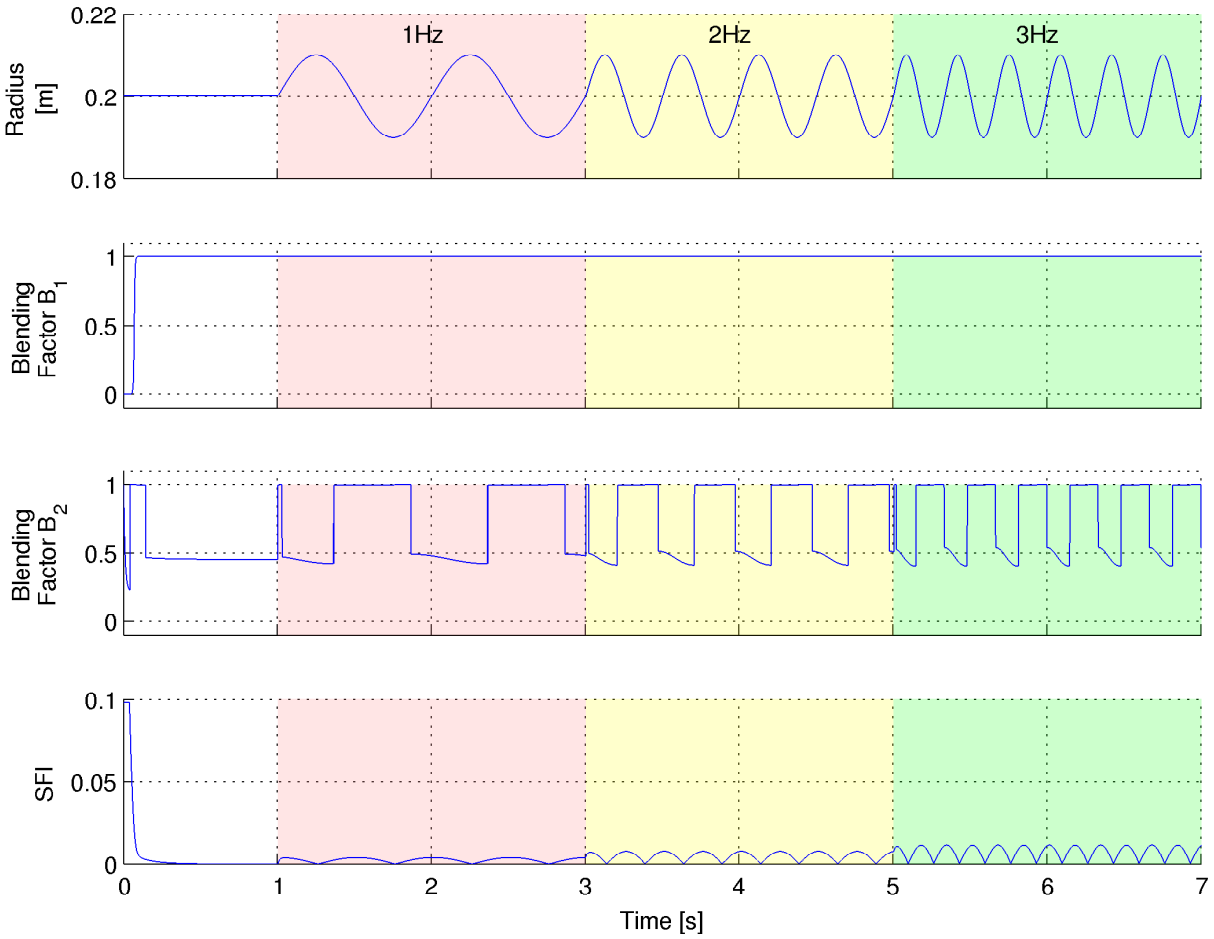


Figure 3.20: Dynamic shape rendering

3.4.2 User Interaction

This section demonstrates the case when haptic interaction takes place in the region of one or more nodes. This case can be understood as dynamic shape rendering with the special feature that specific nodes are controlled in a node positioning task in order to avoid unrealistic haptic impressions, when the shape deforms while touching the object. We simulated the behavior of the FO when a subject touches the FO at node 3 and simultaneously at node 3, 4 and 5. At the beginning of the simulation the FO is in a flat configuration rendering a flat surface ($f(x,y,z) = z$). Figure 3.21 and 3.22 show simulation results when a force acts first on one node only (node 3, $t = 2\,\text{s} \ldots 4\,\text{s}$) and then on three nodes (node 3,4 and 5, $t = 6\,\text{s} \ldots 8\,\text{s}$), respectively. The forces act along the negative z-axis of the global coordinate system. To simulate the shape deformation we use a simple deformation model. The amplitude of the

deformation $\hat{A}$ is calculated with the help of a spring model: $\hat{A} = cf$, with spring constant $c = 1\ \frac{\mathrm{N}}{\mathrm{mm}}$ and force f. The deformation in the surrounding region of the point of interaction is defined by a 2D-Gaussian function $A(x,y) = \hat{A}\exp\left(-\left(\frac{(x-x_0)^2}{2\sigma_x^2} + \frac{(y-y_0)^2}{2\sigma_y^2}\right)\right)$, with (x_0, y_0) the point of interaction on the flat surface. We chose $\sigma_x = \sigma_y = \hat{A}$ for haptic interactions at one point (node 3) and $\sigma_x = \hat{A}$, $\sigma_y = \infty$ for haptic interactions on multiple points (node 3, 4 and 5), which can be interpreted as a line contact. More complicated deformation models could be implemented but would exceed this thesis. Figure 3.21 shows results when all nodes are controlled in a Shape Forming task for the whole duration of the simulation. The flat surface deforms in consequence of the interaction force. One can see that the SFI is low when the shape does not change ($t = 3$ s, $t = 7$ s). Figure 3.23 shows the 3D-plot of the FO adapting to the changed shape at $t = 3$ s and $t = 7$ s, while Figure 3.24 shows the xz-view at $t = 7s$. One can see that the nodes where the interaction takes place change orientation and slide on the virtual surface in order to render the shape by fulfilling the Shape Forming task. In case of a point force on node 3 the device is rotating using the 25th DoF mentioned before in order to better approximate the shape. These movements can be felt by the subject in the form of shear forces and are undesired. In order to avoid such undesired haptic impressions we control the nodes where the haptic interaction takes place to exact positions (Node Positioning task) such that the corresponding nodes follow the point of haptic interaction with no change in orientation (cf. 3.2.9). The remaining nodes are controlled in the Nullspace of these nodes by the Shape Forming task. The result of a simulation controlling specific nodes to exact positions is shown in Figure 3.22, where $SFI(S)$ is the Shape Forming Index of nodes controlled in the Shape Forming task and takes into account the errors of these nodes only. $SFI(P)$ is the Shape Forming Index of the exactly controlled nodes (Node Positioning task). If no haptic interaction takes place ($f = 0$ N), all nodes are controlled by the Shape Forming task and thus, $SFI(P) = 0$. As mentioned before the FO has 25 DoF and 8 nodes are controlled to render the shape. This means that all nodes can be controlled by the Shape Forming task, but when one or more nodes are controlled with the help of the Node Positioning task (6 DoF per node), the remaining DoF are not sufficient to properly control the remaining nodes by the Shape Forming task (3 DoF per node). When controlling all nodes in the Shape Forming task, $SFI(S)$ is zero or decreases towards zero. In Figure 3.22 $SFI(S)$ decreases slowly because B_2 has a low value. As soon as haptic interaction takes place at node 3 ($t = 2$ s$\ldots$4 s), this node is controlled in the Node Positioning task. This task is fulfilled as can be seen from the low $SFI(P)$. On the other hand, the $SFI(S)$ becomes worse because not enough DoF are left for the remaining nodes. The $SFI(S)$ becomes even worse if 3 nodes ($t = 6$ s$\ldots$8 s) are controlled by the Node Positioning task. Figure 3.25 shows the corresponding 3D-plots of the FO adapting to the deformed shape, while Figure 3.26 shows the xz-view at $t = 7s$.

3.4.3 Polygon Sets

In this section, we show results of using polygon sets as shape-input instead of implicit surfaces. The algorithm described in (3.5) uses distances d_i from nodes i to the object and normal vectors $\boldsymbol{n}_i$ at position $\boldsymbol{p}_i$, which are determined at every iteration using the method described in Section 3.3. Thus, polygon sets can be handled similar to implicit surfaces. For

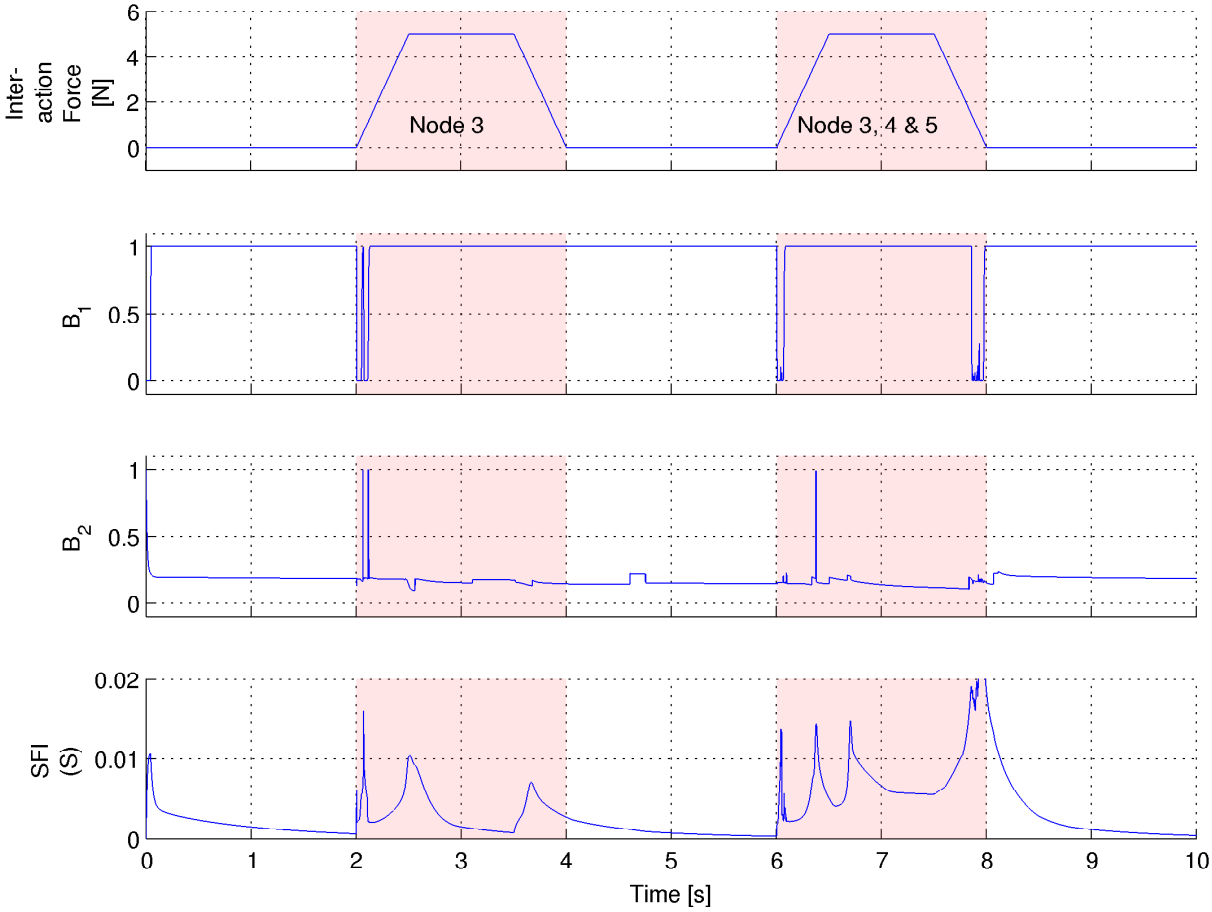


Figure 3.21: Shape rendering of a compliant object with all nodes controlled in a Shape Forming task

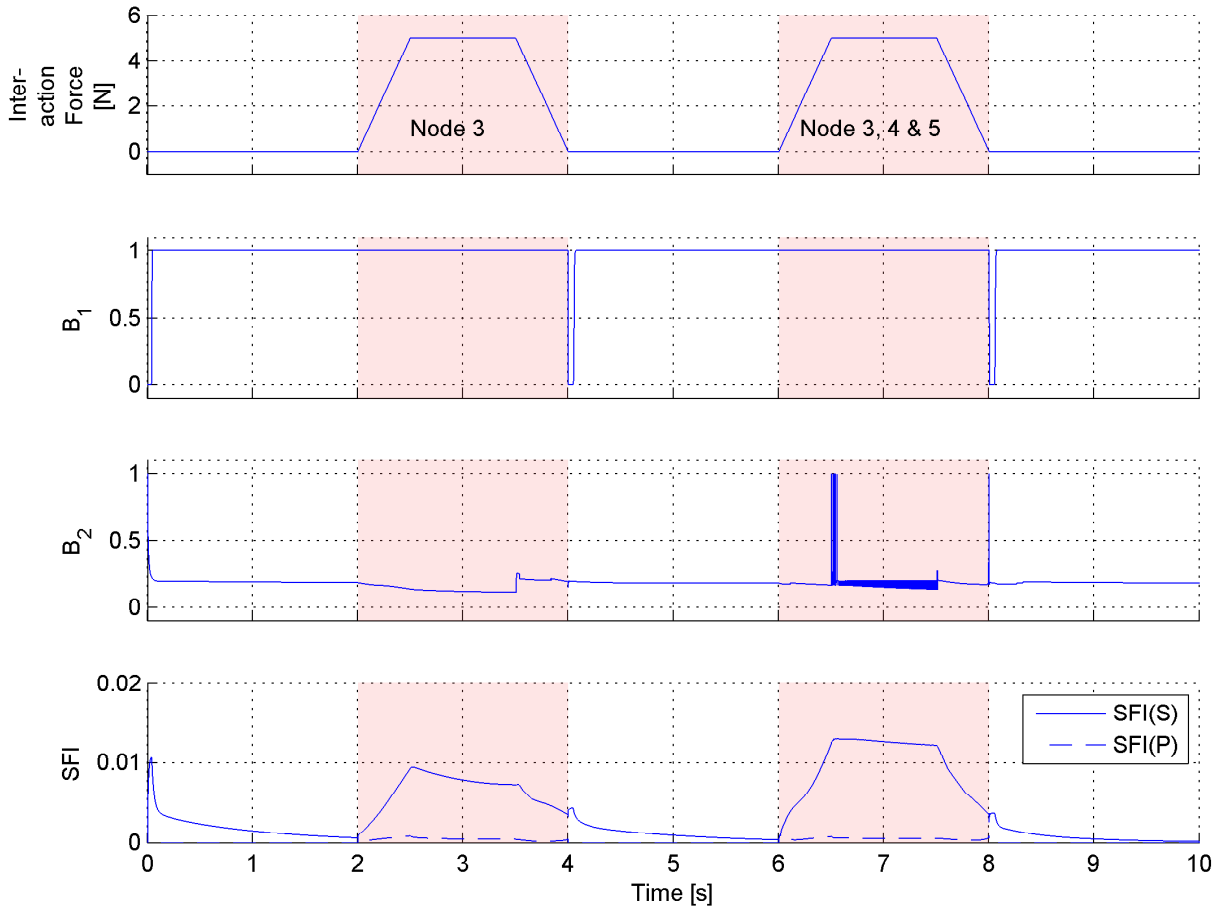


Figure 3.22: Shape rendering of a compliant object with some nodes controlled in a Node Positioning task

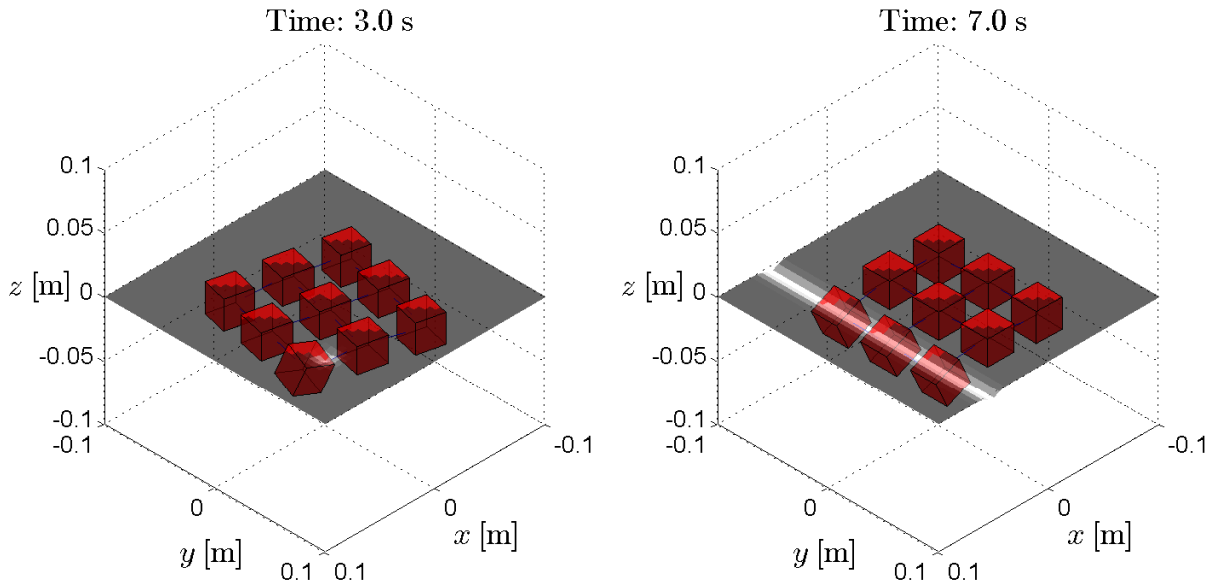


Figure 3.23: FO adapting to deformations when all nodes are controlled in a Shape Forming task (3D-view)

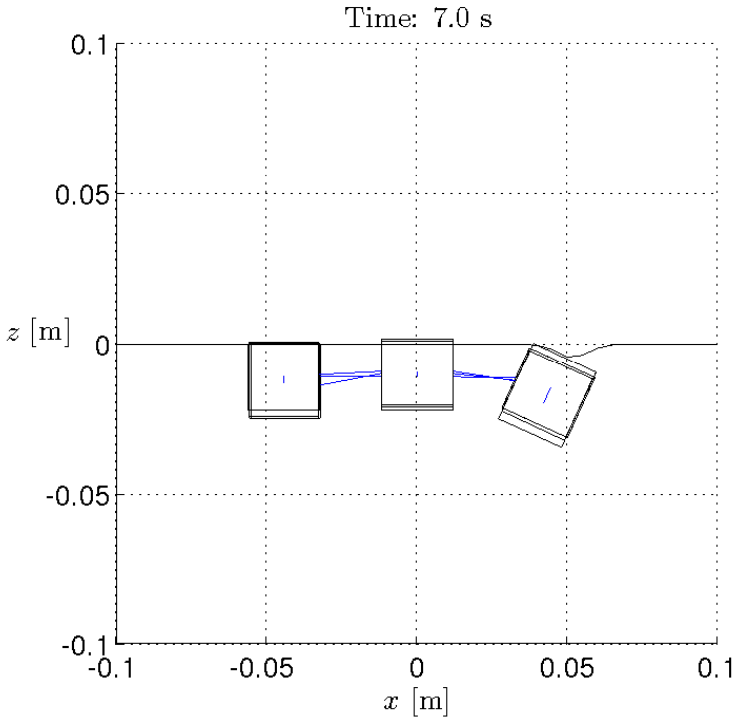


Figure 3.24: FO adapting to deformations when all nodes are controlled in a Shape Forming task (xz-view)

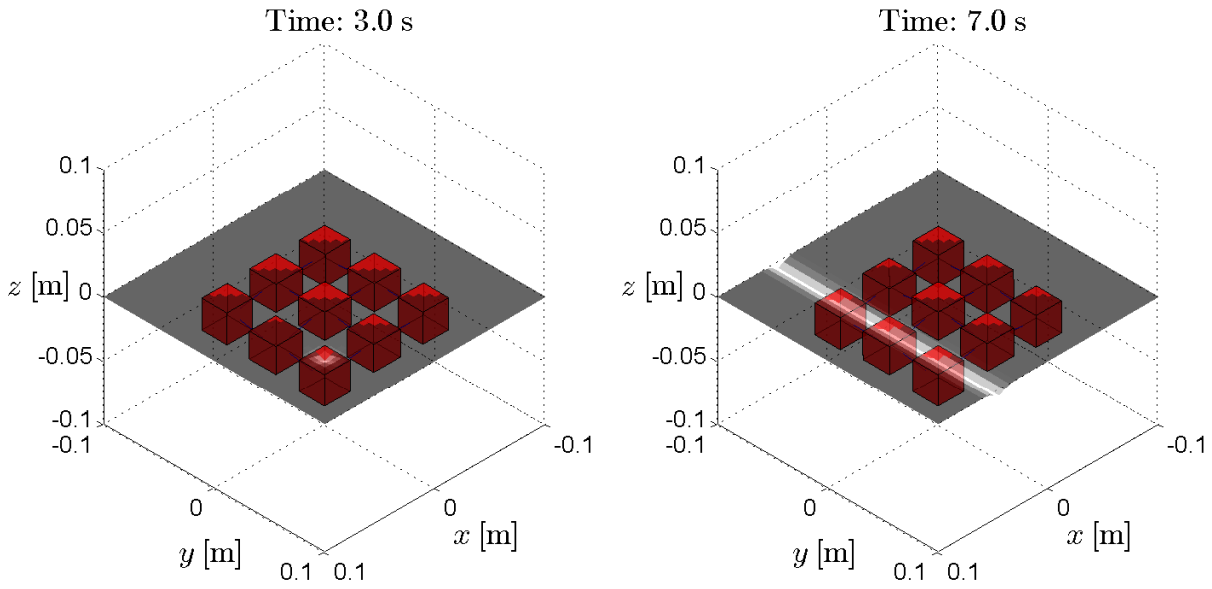


Figure 3.25: FO adapting to deformations when some nodes are controlled in a Node Positioning task (3D-view)

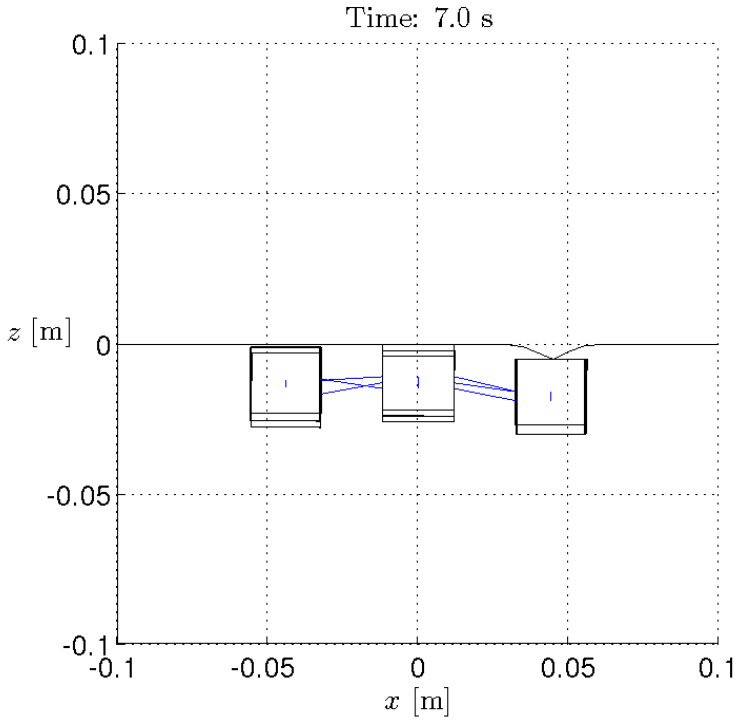
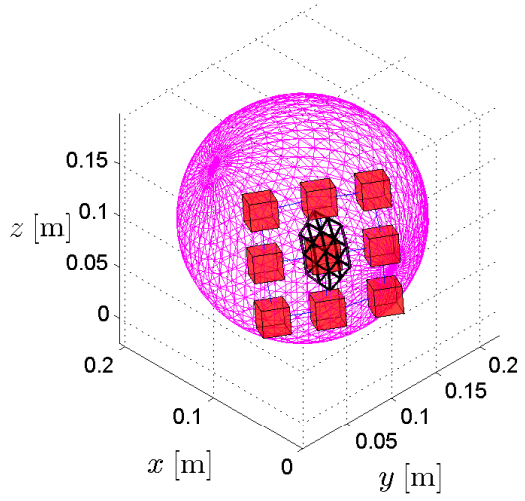


Figure 3.26: FO adapting to deformations when some nodes are controlled in a Node Positioning task (xz-view)

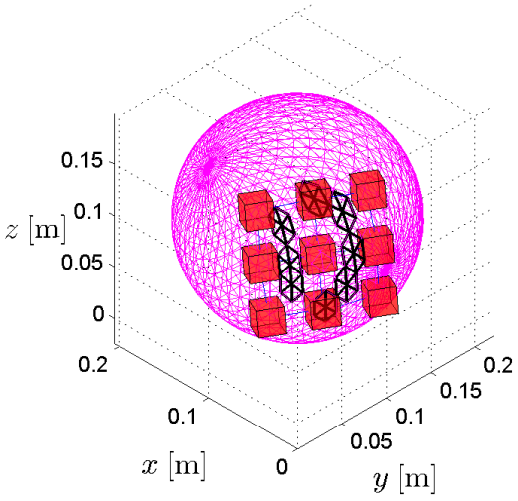
compliant objects, displacements of vertices can be calculated (e.g. FE models). Please note that remeshing must be avoided as this would lead to a recalculation of the offline dataset, which would harm the real-time capability of the algorithm. Node 1 is placed at the center of the virtual surface area, which is to be rendered with the FO. Nodes 2 to 9 (free nodes) are then controlled to be on the surface with the corresponding orientation. At the beginning of the simulation the FO is in a flat configuration. The algorithm described in Section 3.3 requires a meaningful choice of triangles for initialization of the algorithm. This choice has to be made for all free nodes of the FO. We chose the vertex, which is closest to node 1, for all free nodes, because this vertex is close to all nodes even if the object represented by the polygon set is large. Another meaningful choice would be the closest vertex to the corresponding node, but these vertices must be determined offline from the whole polygon set beforehand, which is time consuming. Figure 3.27a-3.27f show the simulation of the FO adapting to the polygon set (sphere with diameter $d = 0.2$ m). One can see how the polygons, which are taken into account by the algorithm (highlighted triangles), shift from node 1 towards the triangles closest to the corresponding node. This happens in the first 5 iterations. The FO then adapts to the shape in about 0.5s. The adaptation to the shape takes relatively long because it is driven by the task error $\boldsymbol{e}_S$ only with a small gain K_S ($K_{Sp} = 100$, $K_{So} = 50$).
Figure 3.28 and 3.29 show the FO adapting to a hexagonal object subscribed by a polygon set with relatively large polygons. One can see that the FO makes a small error due to reached joint limits but renders the object as good as possible. The free nodes position themselves on two neighboring faces.

3.5 Conclusion

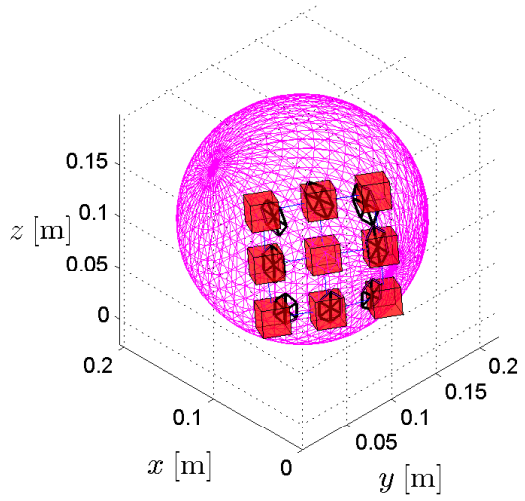
This chapter presented a method for controlling SRIs. In order to approximate a shape, the nodes of the shape rendering interface have to be positioned on the virtual surface and oriented into the normal direction of the surface. This was achieved using differential kinematics within a hierarchical framework, which considered three different tasks: A Shape Forming task as well as two constraint tasks taking into account closed-loop constraints as well as constraints resulting from user interaction. Joint limits and singularities were handled by a weighted-damped least-square Pseudoinverse. Two surface descriptions and their desired proximity queries for haptic rendering were introduced: Implicit surfaces as well as polygon sets. A series of simulations provided insights into the proposed algorithms based on the kinematics of the FO presented in chapter 2 with 9 nodes and 24 DoF and showed that the algorithms run in real-time with a sampling rate of 1 kHz.
In future works one has to further consider different shape descriptions. In this chapter, shapes were described by either implicit functions or polygon data sets. In case of the polygon data set triangular polygons were considered. This is a common description but different polygons must be addressed in future works. Furthermore, shape descriptions in the form of splines must be analyzed because such a description is common in many CAD-software tools.



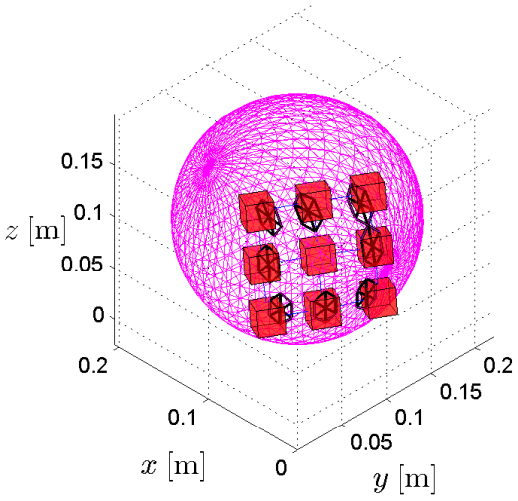
(a) 1st time step



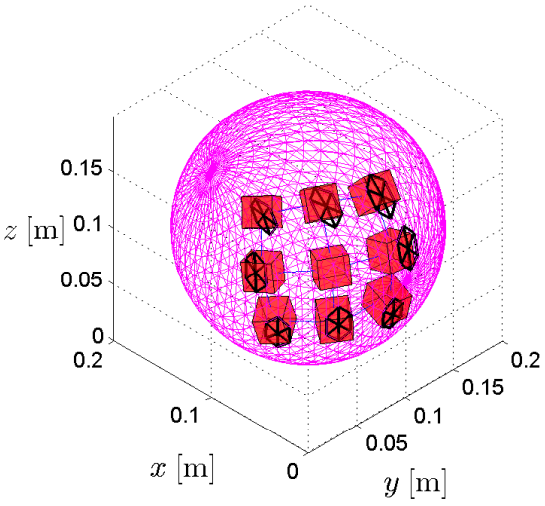
(b) 3rd time step



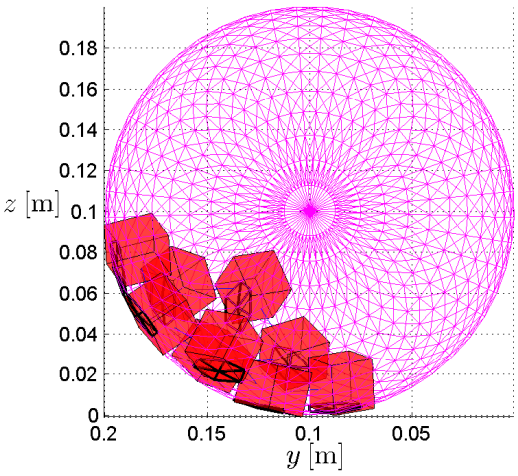
(c) 5th time step



(d) 7th time step



(e) 600th time step



(f) 600th time step

Figure 3.27: FO adapting to a polygon set

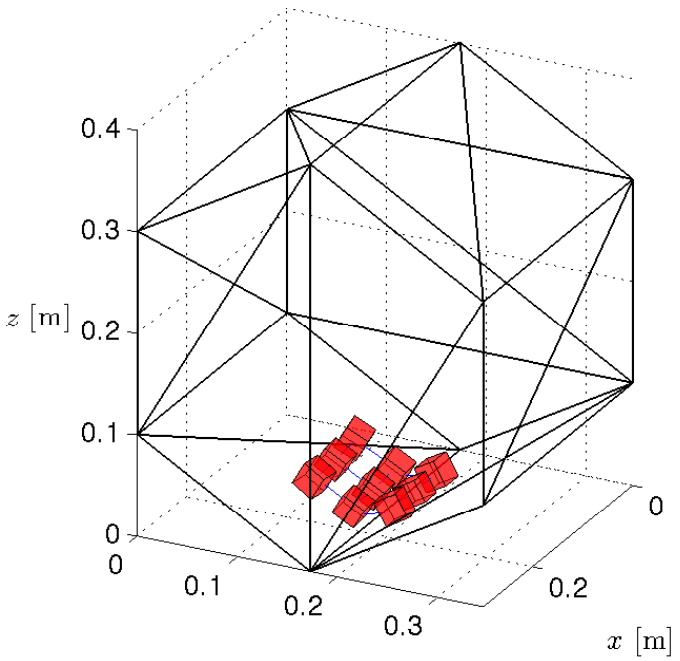


Figure 3.28: FO adapting to a hexagonal object (3D-view)

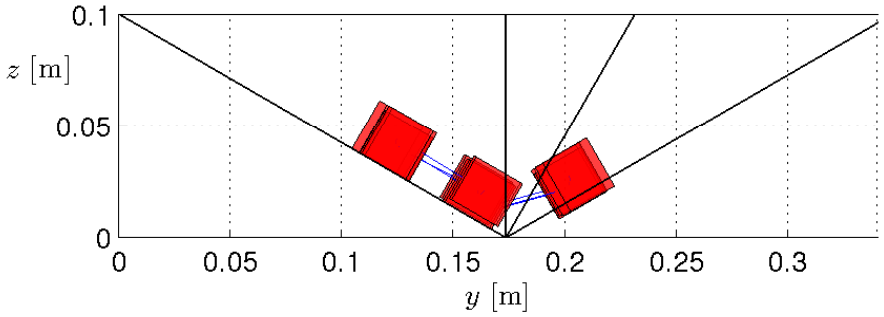


Figure 3.29: FO adapting to a hexagonal object (yz-view)

4 Actuator Reduction for Shape Rendering Interfaces

4.1 Problem Description

Mechanisms with a high amount of DoF, like shape rendering interfaces (SRIs), come along with a high number of actuators because, assuming we are not dealing with an underactuated system, every single DoF must be actuated. We call these actuated joints *active joints* φ_a of the mechanism. In case of a parallel kinematics we also have unactuated joints in the mechanism, which we call *passive joints* φ_p. In this chapter, we analyze a possible actuator reduction for parallel kinematic SRIs. When considering SRIs, one aspires to a mechanism with a high resolution covering a large interaction area. This means that the amount of required actuators grows immensely when pursuing these goals and actuation becomes impractical. In order to reduce the amount of required actuators, sequentially actuating active joints with one actuator only, which can be connected and unconnected to the corresponding joints, represents a practical solution to this problem. In order to not end up with a highly underactuated kinematic system, semi-active actuators (breaks) can be included in the mechanism so that the joints, which are not connected to the actuator remain in their current position. By doing so, the needed space and costs for the actuators can be reduced immensely. On the other hand, actuating the active joints cannot be done without considering passive joints. In other words, when actuating an active joint, joint limits of the passive joints can be reached and thus, a switch to another active joint might be required. In this chapter, we investigate a (sub-)optimal switching sequence, which transfers a mechanism from an initial configuration into a target configuration in minimum time. We assume that the initial and target configuration was previously calculated e.g. with the method described in Chapter 3. Switching from one active joint to another changes the kinematics of the system. Thus, the problem is formally described as a hybrid control problem. The dynamic system model is given by the differential kinematics of the mechanism. To simplify the problem, we, as first attempt, considered a formulation of the problem, which neglects dynamics. By doing so, a permitted solution for the switching sequence is obtained as we assume that we are dealing with a system, which is controlled by a stiff position control. With this assumption the system is able to follow the desired value of the active joint with no or only small errors and the obtained solution will be close to a solution, which also consideres the dynamics of the system. Taking the dynamics into account would namely complicate the problem significantly. Further, an exact model is difficult to obtain. For an exact model, a highly nonlinear model description (friction, couplings, etc.) would have to be developed.

4.2 Hybrid Optimal Control and Heuristic Search

Hybrid control deals with systems where continuous and discrete states and inputs are mixed. Examples for hybrid systems are robots where the system dynamics discretely changes due to the grasping of an object or, automatons where the system dynamics changes due to reached thresholds, etc. An optimal hybrid control minimizes a cost function with regard to multiple constraints (system model, etc.). Several works exist, which extend Pontryagin's maximum principle [63] and the method of dynamic programming for hybrid systems [64], [65], [66] and [67]. Several procedures like mixed-integer programming [68], value function approaches [69] and the hybrid maximum principle [65] have been proposed, to solve hybrid optimization problems. Approaches for subclasses, like switched linear, affine or quadratic dynamics are also known [70], [71], [72], [73]. Most of these concepts assume that a discrete switching sequence is known a priori and solving the optimization problem is to be understood as finding the optimal switching states and times. On the other hand, in the work of Shaikh and Caines ([67] [74] [75]) an optimal switching sequence is identified. The procedure can be regarded as finding the optimum switching pair to transfer a state from one position in the state space to another with one switch. Hereby all possible pairs of positions in the state space are calculated a priori. When aiming for an optimal switching sequence, a combinatorial graph search must be performed in most cases. Approaches of reducing the computational effort like e.g. branch-and-bound methods are typically employed in such situations. [76], [77], [78]. Branch-and-bound methods are a type of heuristic search methods. A wide overview on heuristic search methods is given in [79].
In the following, the hybrid optimization problem is formulated for SRIs. It will be shown, how the problem described in Section 4.1 can be simplified to a purely discrete problem and thus, the hybrid optimization problem can be reduced to a combinatorial graph search problem, which can be efficiently solved with branch-and-bound methods.

Hybrid System Model

In [80] a hybrid dynamic system is defined as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{q}, t) \qquad \text{if} \quad s_i(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{q}, \boldsymbol{v}, t) \neq 0, \tag{4.1}$$

$$\begin{bmatrix} \boldsymbol{x}^+(t) \\ \boldsymbol{q}^+(t) \end{bmatrix} = \boldsymbol{\phi}_i(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{q}, \boldsymbol{v}, t) \quad \text{if} \quad s_i(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{q}, \boldsymbol{v}, t) = 0, \tag{4.2}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{q}, \boldsymbol{v}, t). \tag{4.3}$$

The continuous dynamics of the hybrid system is described in (4.1), while the discrete dynamics is given in (4.2). The hybrid state $\boldsymbol{\xi}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{q}(t) \end{bmatrix}$ is formed by the continuous state $\boldsymbol{x}(t) \in \boldsymbol{X} \subseteq \mathbb{R}^n$ and the discrete state $\boldsymbol{q}(t) \in \boldsymbol{Q} \subseteq \mathbb{N}^s$. Inputs to the system are the continuous control input $\boldsymbol{u}(t) \in \boldsymbol{U} \subseteq \mathbb{R}^m$ and the discrete control input $\boldsymbol{v}(t) \in \boldsymbol{V} \subseteq \mathbb{N}^k$. The transition $\boldsymbol{\phi}_i$ becomes active if the discontinuous hypersurface $s_i = 0$. This is equivalent with a state reset and, if at time t_s the hypersurface i is hit ($s_i(t_s) = 0$), the hybrid state is reset to $\boldsymbol{\xi}(t_s) = \begin{bmatrix} \boldsymbol{x}^+(t_s) \\ \boldsymbol{q}^+(t_s) \end{bmatrix}$. The hybrid system output $\boldsymbol{y} \in \boldsymbol{Y} \subseteq \mathbb{R}^b \times \mathbb{R}^c$ is given by (4.3).

Optimization Cost Function

In [81] the optimization cost function of a hybrid optimization problem, which is to be minimized (or maximized) is defined as follows:

$$J = \theta(\boldsymbol{x}(t_e), t_e) + \sum_{r=1}^{N-1} c_r(\boldsymbol{x}_r, \boldsymbol{v}_r, t_r) + \int_0^{t_e} \phi(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{q}, \boldsymbol{v}, t) dt \tag{4.4}$$

In (4.4) $\theta(\boldsymbol{x}(t_e), t_e)$ specifies the costs for the target state $\boldsymbol{x}_e = \boldsymbol{x}(t_e)$. Switching costs $c_r(\boldsymbol{x}_r, \boldsymbol{v}_r, t_r)$ appear for any of the $N-1$ switches, when passing N systems. The continuous costs, which are described by $\phi(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{q}, \boldsymbol{v}, t)$, occur during the whole time period from $t_0 = 0$ to t_e. The optimization problem is formulated as follows:

$$\begin{aligned} & \min_{\boldsymbol{u},\boldsymbol{v}} \quad J \\ s.t. \quad & \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{q}, t), \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ & \boldsymbol{g}(\boldsymbol{x}(t_e), t_e) = \boldsymbol{0}, \\ & \boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) \leq \boldsymbol{0}. \end{aligned} \tag{4.5}$$

In (4.5) multiple boundary conditions must be taken into account. Beside the continuous system dynamics the initial state $\boldsymbol{x}_0$ and the target condition $\boldsymbol{g}(\boldsymbol{x}(t_e), t_e)$ must be met. Furthermore, inequality constraints $\boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t)$ can be defined for the optimization problem.

4.3 Time and Sequence Optimization for Hybrid Shape Rendering Interfaces

In this section the problem described in Section 4.1 is formulated as a hybrid optimization problem, following the framework of Section 4.2. Especially the boundary conditions as well as the cost function are defined in the following subsections.

4.3.1 Hybrid Model of Shape Rendering Interface

The continuous system model describes the change of the continuous system states $\dot{\boldsymbol{x}}$ with respect to the continuous states $\boldsymbol{x}$ itself and the continuous input $\boldsymbol{u}$. As mentioned before we neglect the dynamics of the system, because we assume that the system is controlled by a stiff position control. As we need a mapping of the continuous state on its derivative, we use the differential kinematics of the mechanism as system model. Multiple continuous system models q_i to describe the current kinematics of the SRI exist, depending on the currently selected actuation of the device. Since the position of passive joints $\boldsymbol{\varphi}_p$ strongly depends on the position of active joints $\boldsymbol{\varphi}_a$, the system model must describe this dependency. Note that a determined system is assumed, which means that the number of active joint m_a is equal the number of DoF M ($m_a = M$). The dependency of all joints $\boldsymbol{\varphi}$ (passive joints $\boldsymbol{\varphi}_p \in \boldsymbol{X}_p \subseteq \mathbb{R}^{n_p}$ included) on active joints $\boldsymbol{\varphi}_a \in \boldsymbol{X}_a \subseteq \mathbb{R}^{n_a}$ is formulated by:

$$\boldsymbol{\varphi} = \begin{bmatrix} \boldsymbol{\varphi}_a^T & \boldsymbol{\varphi}_p^T \end{bmatrix}^T = \boldsymbol{f}_{pa}(\boldsymbol{\varphi}_a) \tag{4.6}$$

As we are dealing with parallel kinematics, the dependency given by (4.6) cannot be described analytically in most cases, especially if the mechanism has a high amount of DoF. On the other hand, the differential dependency $\hat{\boldsymbol{f}}_{pa}$ (4.7) can be stated more easily.

$$\dot{\boldsymbol{\varphi}} = \begin{bmatrix} \dot{\boldsymbol{\varphi}}_a^T & \dot{\boldsymbol{\varphi}}_p^T \end{bmatrix}^T = \hat{\boldsymbol{f}}_{pa}(\boldsymbol{\varphi}_a, \boldsymbol{\varphi}_p, \dot{\boldsymbol{\varphi}}_a) \tag{4.7}$$

The joint angles of the mechanism define the continuous state of the hybrid system $\boldsymbol{\varphi} = \begin{bmatrix} \boldsymbol{\varphi}_a^T & \boldsymbol{\varphi}_p^T \end{bmatrix}^T := \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_a^T & \boldsymbol{x}_p^T \end{bmatrix}^T$, while the (scalar) discrete state q is defined by the actuated active joint. The (scalar) continuous system input u is defined by the velocity of the actuator, which drives the actuated active joint. The (scalar) discrete input v is defined by the actuated joint itself. Note that the discrete state q is directly correlated to the discrete input v. In other words, a change of the discrete input changes the discrete state so that $q = v$.

Differential Dependency of Passive Joints on Active Joints

As mentioned before a differential dependency $\hat{\boldsymbol{f}}_{pa}$ of passive joints $\boldsymbol{\varphi}_p = \boldsymbol{x}_p$ on active joints $\boldsymbol{\varphi}_a = \boldsymbol{x}_a$ is used as dynamic model considered for optimization. In the following, it is shown how a differential kinematic solution (4.7) can be determined with the help of a hierarchical task space projection.
The dependency of passive joints on active joints is determined by the closed loops of the parallel kinematics. When actuating one or more active joints, the passive joints always adapt so that these loop constraints are fulfilled. Figure 4.1 shows an example kinematics with $M = 6$ DoF. The mechanism contains $m_a = 6$ active and $m_p = 3$ passive joints. The mechanism is statically determined as $M \stackrel{!}{=} m_a$. As for a serial kinematics it is easy to identify the forward kinematics in an analytical way, every loop of the mechanism is virtually cut free, and thus, two serial kinematics per loop are gained (a and b)(cf. chapter 3). Figure 4.2 shows the previous example with a virtual cut at node 5. The forward kinematics, which describes the position and orientation of the cut using the kinematics a or b can be determined in an analytical way. In the following we want to use the algorithm of hierarchical Nullspace projection (cf. Chapter 3 and [25]) to determine the differential kinematics $\hat{\boldsymbol{f}}_{pa}$ of the mechanism. Thus, we define two tasks: *Loop Closure* and *Active Joint Positioning*. The Loop Closure task controls the serial kinematics (a and b) so that the loops, which were virtually cut free, are closed. This task must be fulfilled at every time and thus, gains the highest priority in the task hierarchy. The Active Joint Positioning task is executed on the Nullspace of the Loop Closure task and its goal is to drive the active joint angles to its desired values. In order to fulfill the previously introduced tasks, the passive joints must adapt so that both tasks are fulfilled. Note, that the Node Positioning task and the Shape Forming task, which were introduced in Section 3.2.3 are not considered here. In this chapter, the method of hierarchical task-space projection is solely used to gain a dependency of passive joints on active joints.

The hierarchical Nullspace projection is described in detail in Section 3.2.1 (3.5). In order to use the algorithm, Jacobians and task velocities must be defined for each task. These definitions are made in the following.

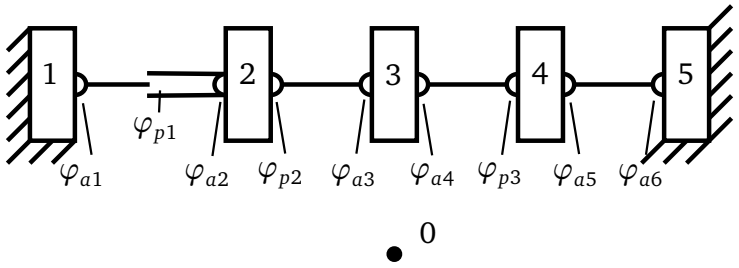


Figure 4.1: Parallel kinematics with 6DoF

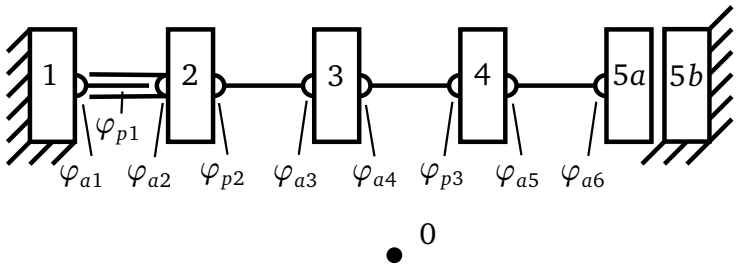


Figure 4.2: Parallel kinematics with 6 DoF (cut at node 5)

Task 1 (Loop Closure): The Loop Closure task is the most important task in the hierarchy as it guarantees the fulfillment of the loop closure constraints. A detailed description about the Loop Closure task was given in Section 3.2.4. The most important aspects of the Loop Closure task are recapitulated in the following. As mentioned before, the serial forward kinematics of the open loop j ($\boldsymbol{f}_{Cja}$ and $\boldsymbol{f}_{Cjb}$) can be calculated analytically. By subtracting $\boldsymbol{f}_{Cja}$ from $\boldsymbol{f}_{Cjb}$ the relative position and (orientation) of the two sides of the cut $\boldsymbol{f}_{Cj}$ is obtained (cf. (3.11)). The time derivative of $\boldsymbol{f}_{Cj}$ leads to the Jacobian of the loop closure task $\boldsymbol{J}_C = \boldsymbol{J}_1$ (cf. (3.12)). The task velocity $\dot{\boldsymbol{x}}_{Cj}$ is the relative Cartesian velocity (position and orientation) of the two sides of the cut of loop j. As the relative position (and orientation) must be zero at all times the task velocity must be $\dot{\boldsymbol{x}}_{Cj} = \boldsymbol{0}$. Note that the index is $j = 1 \ldots l$ with l the number of loops.

Task 2 (Active Joints): The Jacobian $\boldsymbol{J}_2 = \boldsymbol{J}_A$ of the Active Joint Positioning task shows the dependency of active joints $\boldsymbol{\varphi}_a$ on all joints $\boldsymbol{\varphi} = \begin{bmatrix} \boldsymbol{\varphi}_a^T & \boldsymbol{\varphi}_p^T \end{bmatrix}^T$:

$$\dot{\boldsymbol{\varphi}}_a = \boldsymbol{J}_2(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}} = \boldsymbol{J}_A(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}}. \tag{4.8}$$

In the cut free state, the active joints have no effect on the passive joints and thus, $\boldsymbol{J}_A$ is composed of an identity- and a zero-matrix:

$$\boldsymbol{J}_A = \left[\begin{array}{c:c} \underset{[m_a \times m_a]}{\boldsymbol{I}} & \underset{[m_a \times m_p]}{\boldsymbol{0}} \end{array} \right]. \tag{4.9}$$

The according Pseudoinverse $\boldsymbol{J}_A^*$ is given by the transposed matrix:

$$\boldsymbol{J}_A^* = \boldsymbol{J}_A^T \boldsymbol{J}_A \boldsymbol{J}_A^T = \boldsymbol{J}_A^T. \tag{4.10}$$

The algorithm (3.5) executes tasks of lower priority on the Nullspace of higher priority tasks. The adjusted Pseudoinverse is calculated by multiplying the Pseudoinverse with the

Nullspace of the higher priority task. Thus, the needed Pseudoinverse for the Active Joint Positioning task is calculated as follows:

$$\boldsymbol{J}_{AN} = \boldsymbol{J}_A \boldsymbol{N}_C \text{ and } \boldsymbol{J}^*_{AN} = \boldsymbol{N}^*_C \boldsymbol{J}^*_A, \tag{4.11}$$

$$\dot{\boldsymbol{\varphi}} = \begin{bmatrix} \dot{\boldsymbol{\varphi}}_a^T & \dot{\boldsymbol{\varphi}}_p^T \end{bmatrix}^T = \boldsymbol{J}^*_{AN} \dot{\boldsymbol{\varphi}}_a. \tag{4.12}$$

Note that neither $J_1 = J_C$ nor $J_2 = J_A$ use damping or weighting, because it would falsify the kinematics, which describes the dependency of passive joints on active joints. The task velocity $\dot{\boldsymbol{x}}_2 = \dot{\boldsymbol{x}}_A$ is given by the velocity of the active joints $\dot{\boldsymbol{\varphi}}_a$, which is a vector of the length m_a.

Continuous Dynamics

Equation (4.12) defines the required differential dependency of passive joints on active joints. Thus, the continuous system model is:

$$\dot{\boldsymbol{\varphi}} = \hat{\boldsymbol{f}}_{pa}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}_a) = \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{\varphi}}_a) = \boldsymbol{J}^*_{AN}(\boldsymbol{x}) \dot{\boldsymbol{\varphi}}_a. \tag{4.13}$$

In (4.13) $\dot{\boldsymbol{\varphi}}_a$ is considered the continuous system input, which is a vector with m_a elements. As, per definition, we can actuate one active joint at a time only, $\dot{\boldsymbol{\varphi}}_a$ is in the following described with the help of u and v.

Discrete Dynamics

The discrete input v is defined on the finite discrete input space $v \in \mathbb{N} = \{0, 1, 2, \ldots, m_a\}$, with m_a the number of active joints. As only one active joint can be actuated at a time per definition, the continuous input u must appear at the corresponding position in $\dot{\boldsymbol{\varphi}}_a$, while the other elements must be zero. Thus $\dot{\boldsymbol{\varphi}}_a$ is described with u and $v = q$ in the following way:

$$\dot{\boldsymbol{\varphi}}_a = \begin{bmatrix} 1 - \operatorname{sgn}(|1 - v|) \\ 1 - \operatorname{sgn}(|2 - v|) \\ \vdots \\ 1 - \operatorname{sgn}(|n_v - v|) \end{bmatrix} u = \begin{bmatrix} 1 - \operatorname{sgn}(|1 - q|) \\ 1 - \operatorname{sgn}(|2 - q|) \\ \vdots \\ 1 - \operatorname{sgn}(|n_v - q|) \end{bmatrix} u. \tag{4.14}$$

With this definition u only appears at the position of $\dot{\boldsymbol{\varphi}}_a$ of the corresponding active joint and (4.12) becomes:

$$\dot{\boldsymbol{\varphi}} = \dot{\boldsymbol{x}} = \boldsymbol{J}^*_{AN} \dot{\boldsymbol{\varphi}}_a = \boldsymbol{J}^*_{AN}(\boldsymbol{x}) \begin{bmatrix} 1 - \operatorname{sgn}(|1 - q|) \\ 1 - \operatorname{sgn}(|2 - q|) \\ \vdots \\ 1 - \operatorname{sgn}(|n_v - q|) \end{bmatrix} u = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{q}, t) \tag{4.15}$$

Transitions

In the case of the SRI we have control over the switching at all times, as switching can be dictated by the discrete control input v. Thus, the transition has a simple form:

$$s_i(v) = p - v \quad \text{with} \quad p \in \boldsymbol{Q}. \tag{4.16}$$

This description means that a switch to system p (active joint φ_{ap}) occurs if $v = p$. When switching from one system to another, the continuous state $\boldsymbol{x}$ does not change, but only the discrete state q changes to the new system:

$$\begin{bmatrix} \boldsymbol{x}^+(t) \\ q^+(t) \end{bmatrix} = \boldsymbol{\phi}_i(v,t) = \begin{bmatrix} \boldsymbol{x}^-(t) \\ v^+(t) \end{bmatrix} \tag{4.17}$$

Hybrid Optimization of Switching Sequence of Shape Rendering Interface

The previously described hybrid system is to be transferred from an initial hybrid state $\xi_0 = \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{q}_0 \end{bmatrix}$ into a target state $\xi_e = \begin{bmatrix} \boldsymbol{x}_e \\ \boldsymbol{q}_e \end{bmatrix}$, which were previously calculated using e.g. the methods described in Chapter 3. Note that the discrete system state $\boldsymbol{q}_e$ is of minor interest, as any active joint may be connected to the actuator, when the continuous target state $\boldsymbol{x}_e$ is reached. In the following we want to find an optimal switching sequence $Q = \{q_0, q_1, \ldots, q_{N-1}\}$ so that the system is transferred from ξ_0 into ξ_e in minimal time. When applying the sequence Q, N systems are passed, while the systems of Q are not necessarily distinct. Thus, $N-1$ switches occur. We assume that switching from one system to another takes a constant time T_{Switch}. Thus, the cost function, which is to be minimized in the optimization, is defined by:

$$J = \underbrace{\sum_{j=1}^{N-1} T_{Switch}}_{\text{discr. costs}} + \underbrace{\int_0^{t_e} 1 dt}_{\text{cont. costs}}. \tag{4.18}$$

By comparing (4.18) with (4.4) one can see that:

$$\theta(\boldsymbol{x}(t_e), t_e) = 0; \quad c_r = T_{Switch}; \quad \phi = 1 \tag{4.19}$$

The cost function in (4.18) includes continuous costs and discrete costs. The discrete costs are caused by the time, which is required to switch from one system to another. The continuous costs are caused by the time, which is needed to drive from the initial state to the target state. Thus, the cost optimal control must have as least switches as possible on the one hand and on the other hand, drive as fast as possible from one continuous state to another. This means that the cost function can be further simplified making the following assumptions for our simplified scenario: As we consider only one active joint to be actuated (and all other active joints are blocked) at a time, the path of this active joint and the passive joints is predetermined by the state at the switching time, and can be illustrated by a fixed trajectory in the state space. Thus, the time to cover a distance on that trajectory can only be minimized by driving the corresponding active joint with maximum velocity. This can easily be shown

with the help of Pontryagin's maximum principle. Taking a different trajectory in the state space to transfer the state from one state to another is not possible. As this fact holds for all continuous system models q_i, the continuous system input must be set to its maximum value $u = \pm u_{max}$ to minimize the continuous costs of the cost function. Furthermore, one can say that, when using a maximal continuous input at all times, the continuous costs will take a constant value no matter which switching sequence is used. This statement only holds, when the active joints are driven towards their desired values, which is assumed here, in order to further simplify the problem. Driving the active joints away from their target would increase the continuous costs of the cost function and considering the problem as discrete problem would not be allowed in this case. Thus, the cost function is simplified to:

$$\hat{J} = \sum_{j=1}^{N-1} T_{Switch}. \tag{4.20}$$

Thus, minimizing the cost function is a matter of finding a switching sequence in which as least switches as possible occur.

4.3.2 Combinatorial Graph Search

As the optimization problem can be reduced to the problem of finding an optimal switching sequence (as least switches as possible), the problem is transformed into a combinatorial graph search optimization. The challenge lies in the cardinality of the graph. As mentioned before switching from one system to any of the other systems can be conducted at any time (at any position in the state space). This fact blows up the graph and the number of branches, which have to be analyzed to find a global optimum. In the following paragraphs a general approach of finding a (sub-)optimal solution in the obtained graph is shown. Based on this approach suboptimal and optimal solutions are discussed.

Graph Search Algorithm

To find a (sub-)optimal solution in a branch tree, Stursberg presented a method that optimizes the discrete and continuous degrees of freedom in a two-stage procedure [77]. The discrete controls are selected by a graph search algorithm and the continuous controls are obtained by embedded nonlinear programming. As the continuous part of the optimization problem examined here is trivial, the graph search algorithm is of interest, which implements the well-known A*-algorithm which is briefly recapitalized in the following paragraph.
A graph G is defined to be a set $\{n_i\}$ of elements called nodes and a set $\{e_{ij}\}$ of directed line segments called arcs [82]. The element e_{pq} is an arc from node n_p to n_q and n_q is a successor of n_p. Each arc has costs c_{ij} associated with it (see Figure 4.3). The purpose of the A*-algorithm is to find the path with minimum costs in the graph, which leads from an initial node n_0 to a target node n_e (e.g. n_5). The costs from the initial node n_0 to the target node n_e are evaluated using an evaluation function $f(n_i)$, which calculates the costs of the path when passing node n_i:

$$f(n_i) = g(n_i) + h(n_i). \tag{4.21}$$

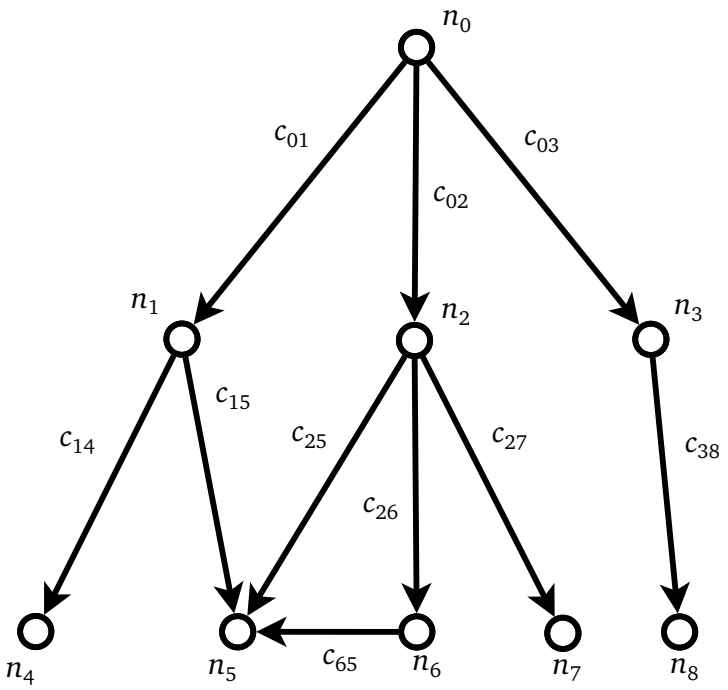


Figure 4.3: Example of a graph

In (4.21) $g(n_i)$ is the actual cost of an optimal path from n_0 to n_i. The function $h(n_i)$ estimates the costs from n_i to the target node n_e, which is done by a lower bound estimation. Note that the graph G is expanded during the run of the algorithm. This means that the graph is not completely available at the beginning, but it is rather expanded during the run of the algorithm. This clarifies why an estimation function $h(n_i)$ is needed. Multiple modes to expand the graph are known: *breadth-first*, *depth-first* and *best-first* [77]. The *breadth-first*-mode selects all nodes of the graph from which the system evolution has to be investigated further. The *depth-first*-mode selects the node from the last iteration, which has minimum cost. And the *best-first*-mode selects the node from the whole graph, which has minimum cost ($\min f(n_i)$). For the following considerations the *best-first*-mode is used.

A*-Algorithm applied to the Time Optimization Problem

In this section the aforementioned A*-algorithm is applied to the combinatorial problem of the SRI. Note that we are dealing with controlled switching and thus, a switch from one system to another can take place at any time and state. Thus, discretizing the state space is a suitable procedure to avoid an infinite cardinality. The highest discretization, which is most sensible in a practical sense, is dictated by the resolution of the encoders measuring the joint angles of the active joints φ_a, which are called *discretization points* in the following. Such a high resolution in most cases still leads to a high cardinality, but for the following derivations, we stick to this resolution, if not specified differently. Let us assume that we deal with a kinematics with m_a active joints and m_p passive joints. Furthermore, $\hat{m}_{a0} \leq m_a$ active joints are initially not in their corresponding desired target state. Thus, the minimum

costs estimated are:

$$h(n_0) = \hat{m}_{a0} T_{switch}. \tag{4.22}$$

The value estimated by (4.22) assumes that no joint limits of the passive joints φ_p are violated when driving the active joints φ_a towards their desired values, which delivers a lower-bound estimation. This estimation can be used at any node in the path:

$$h(n_i) = \begin{cases} \hat{m}_a(n_i) T_{switch} & \text{if } q^- \neq q^+, \\ (\hat{m}_a(n_i) - 1) T_{switch} & \text{else .} \end{cases} \tag{4.23}$$

In (4.23) $\hat{m}_a(n_i)$ determines the number of active joints which have not reached their desired values at node n_i, yet. The second case takes care of paths, which do not include a switch to another system. This occurs if a discrete state position is met and proceeding with the same system is permitted. Figure 4.4 shows node n_i and its possible successors. For the path where no switch to another system takes place ($q = 1$), the lowest cost is estimated by $h(n_i)$. This path will always be preferred by the algorithm as it promises the lowest costs. After defining $h(n_i)$ we have to define $g(n_i)$, which delivers the actual costs at node n:

$$g(n_i) = m_{SW}(n_i) T_{switch}. \tag{4.24}$$

In (4.24) $m_{SW}(n_i)$ is the number of switches conducted to reach node n_i.

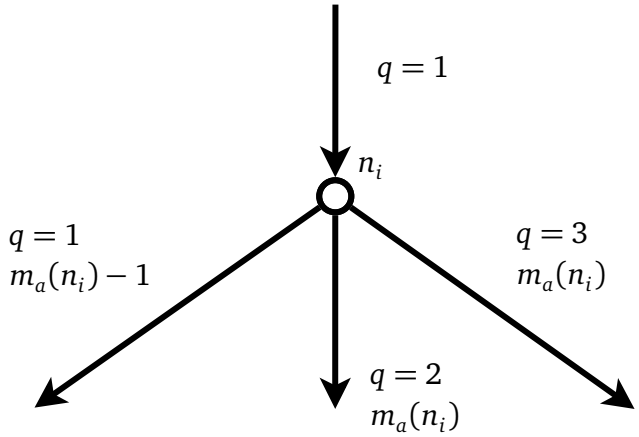


Figure 4.4: Cost estimation at node n_i

Transitions

In 4.3.1 it was stated that switching to any other system is possible at any time. In order to reduce the cardinality arising from this assumption, the transitions are redefined in the following way. A new node/transition arises, whenever one of the following three cases occurs:

1. The desired value of the active joint is reached.
2. A joint limit of one of the passive joints is reached.

3. A discretization point of the active joint is reached.

These three cases are mathematically formulated as:

$$s_i = |\left(\varphi_{aq} - \varphi_{aq,e}\right)| + |(p - v)|, \tag{4.25}$$
$$\text{with } v \in \boldsymbol{V} \mid \left(\varphi_{av} - \varphi_{av,e}\right) \neq 0 \text{ and } p \in \boldsymbol{Q},$$
$$s_i = |\left(\varphi_{aq} - \varphi_{aq,min}\right)\left(\varphi_{aq} - \varphi_{aq,max}\right)| + |(p - v)|, \tag{4.26}$$
$$\text{with } v \in \boldsymbol{V} \mid \left(\varphi_{av} - \varphi_{av,e}\right) \neq 0 \wedge v \neq q \text{ and } p \in \boldsymbol{Q},$$
$$s_i = |\prod_{b=1}^{n_d - 1} \left(\varphi_{aq} - \varphi_{aq,d}(b)\right)| + |(p - v)|, \tag{4.27}$$
$$\text{with } v \in \boldsymbol{V} \mid \left(\varphi_{av} - \varphi_{av,e}\right) \neq 0 \text{ and } p \in \boldsymbol{Q}.$$

In (4.25) - (4.27) φ_{aq} denotes the active joint (system q) and $\varphi_{aq,e}$ is its target state. Each active joint φ_{aq} has a valid range, reaching from the minimum joint angle $\varphi_{aq,min}$ to the maximum $\varphi_{aq,max}$. This range is discretized into n_d discrete sectors. These sectors are separated by the discretization points:

$$\boldsymbol{\varphi}_{aq,d}(b) = \varphi_{aq,min} + b\Delta\varphi_{aq} = \varphi_{aq,min} + b\frac{(\varphi_{aq,max} - \varphi_{aq,min})}{n_d} \quad \text{with } b = 1 \ldots (n_d - 1) \tag{4.28}$$

Figure 4.5 shows the previously introduced kinematics in a start configuration (solid lines) and a target configuration (dashed lines). One can see that in order to drive the configuration to the target, the active joint 4, 5 or 6 can be chosen in the beginning. Figure 4.6 visualizes the possible paths at the initial state. One can see that when choosing active joint 4, no joint limits are violated on the way to its target state. In the case of active joint 5 or 6 the target position for the corresponding active joint cannot be reached, due to joint limit violations. The discretization points $\boldsymbol{\varphi}_{aq,d}$ are shown for a discretization with $n_d = 5$. The joint limits of the passive joints are illustrated by a box.

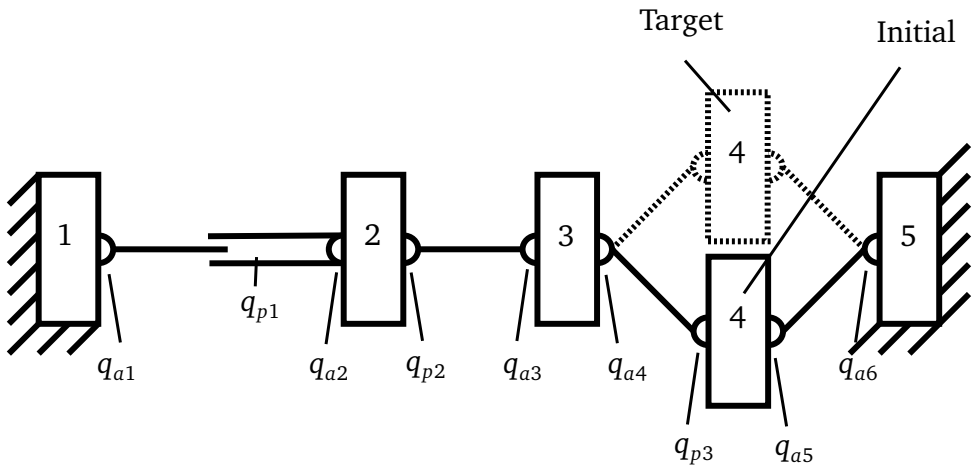


Figure 4.5: Initial and target configuration of example kinematics

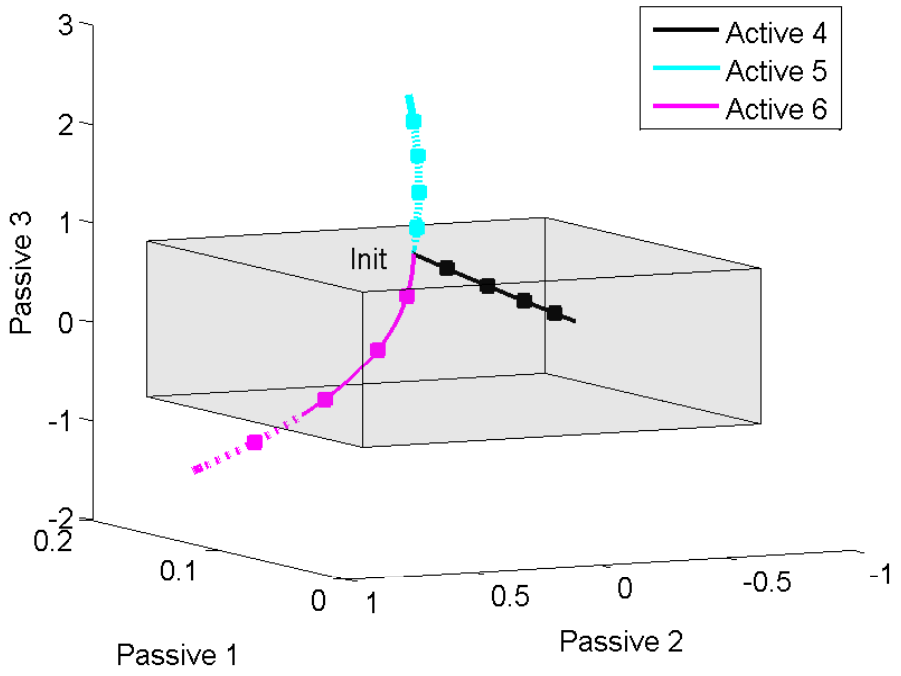


Figure 4.6: Comparison of 3 possible paths in the state space: joint limits are reached when driving active joint 5 or 6 towards their desired values

(Sub-)optimal Solution

As mentioned before, the cardinality of the graph search tree tends to infinity, when all possible permutations are to be considered. Beside that, we know that, if no joint limits are violated, the algorithm will try to avoid switches and thus, switches at the boarder of the allowed passive state space will be favored. This leads us to a description for the transitions, which massively reduces the cardinality of the search tree. We artificially set the transitions so that only case 1 and case 2 (4.25)(4.26) are considered. As, only in rare cases, switching at discretization points leads to an optimal solution, the corresponding transitions are disregarded. By doing so a lot less paths have to be considered while keeping the most promising paths in the search. The following statements can be made:

1. If $m_{SW}(n_e) \leq \hat{m}_{a0} + 1$ a global optimal solution is found
2. If $m_{SW}(n_e) > \hat{m}_{a0} + 1$ a sub-optimal solution is found which might be the global optimum

These statements are motivated by the following chain of thoughts. As the lowest thinkable number of switches is known a priori ($\hat{m}_{a0}$) one can say that $\hat{m}_{a0}$ must be a global optimum. All possible permutations to reach that value are included in the search with the transitions defined before (4.25) (4.26). Thus, if no solution with $\hat{m}_{a0}$ switches is found a solution with $\hat{m}_{a0} + 1$ switches must be a global optimum. As the transitions only include switching if a passive joint angle limit is reached, there might be a solution with less switches (switching somewhere in the state space) in the case when more that $\hat{m}_{a0} + 1$ switches are found with

the graph search. If one wants to give a statement about global optimality, one would have to consider a search with a higher cardinality (discretization points included). This search can be interrupted once a solution of the first case is found. The suboptimal solution $\tilde{m}_{SW}(n_e)$ is used as lower-bound estimation so that branch-and-bound approaches can be used. This avoids, that paths which contain more switches than found within the suboptimal solution ($\tilde{m}_{SW}(n_e)$), are evaluated. A path is pruned as soon as $f(n_i) = g(n_i) + h(n_i) \geq \tilde{m}_{SW}(n_e) T_{switch}$ (cf. (4.21)). As the number of paths, which have to be evaluated to find a solution, is not known a priori, the method presented here cannot run in real-time. On the other hand, the system behavior is well predictable, which means that little time is needed to search large graphs.

4.4 Results

In this section we aim for demonstrating the functionality of the (sub-)optimal search procedure. The mechanism introduced earlier (Figure 4.1) is used to demonstrate the optimization of the switching sequence.
The kinematics is to be transferred from an initial position to a target position. Transitions occur if the joint limit of one of the passive joints is reached or, if the active joint reaches its desired value (no discretization). The A*-algorithm with best-first mode is applied to the optimization problem. In the following, we show examples for the two cases introduced in 4.3.2.

Optimal Solution: $m_{SW}(n_e) = \hat{m}_{a0}$

Figure 4.7 shows the initial and the target configuration of the mechanism. One can see that $\hat{m}_{a0} = 3$ active joints (4,5,6) are not in the target state at the beginning. A solution with $m_{SW}(n_e) = 3$ switches is found with the A*-algorithm ({4 5 6}), which is an optimal sequence. The A*-algorithm generated 4 nodes (initial and target node included), until the solution was found. The path of the passive states is shown in Figure 4.8. One can see that no joint constraints are violated during the state transfer from $\boldsymbol{x}_0$ to $\boldsymbol{x}_e$.

Optimal Solution: $m_{SW}(n_e) = \hat{m}_{a0} + 1$

Figure 4.9 shows a different initial and target configuration of the mechanism. Active joints 1, 2, 5 and 6 are not in the target state at the beginning, which leads us to $\hat{m}_{a0} = 4$. A sequence with $m_{SW}(n_e) = 5$ switches was found ({2 6 5 1 5}). As all potential paths which could lead to $m_{SW}(n_e) = 4$ were included in the search one can say, that an optimal solution was found and a solution with only 4 switches does not exist. The A*-algorithm generated 21 nodes (initial and target node included), until the solution was found. The path of the passive states is shown in Figure 4.10. One can see that no joint constraints are violated during the state transfer from $\boldsymbol{x}_0$ to $\boldsymbol{x}_e$.

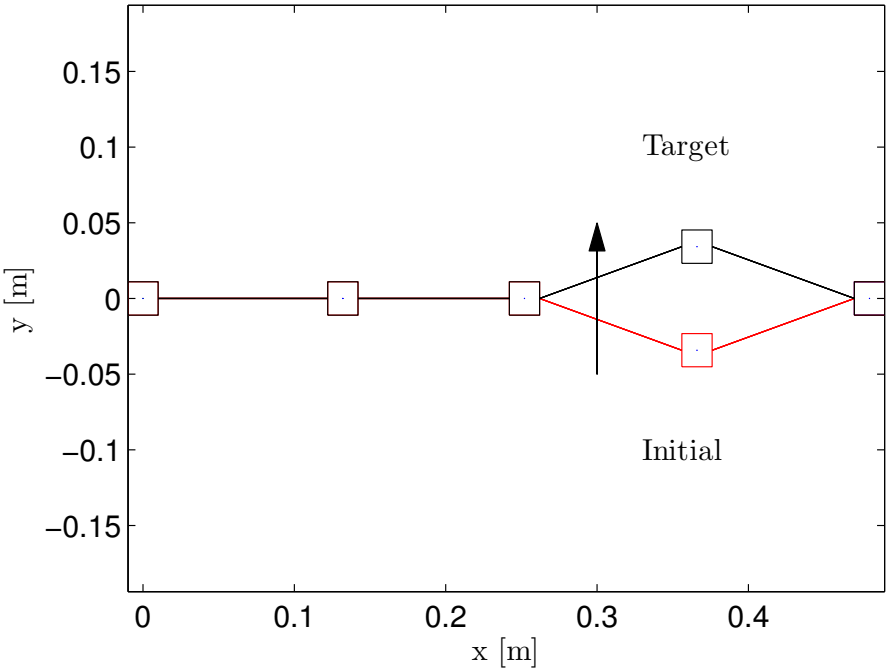


Figure 4.7: Initial and target configuration ($m_{SW}(n_e) < \hat{m}_{a0} + 1$)

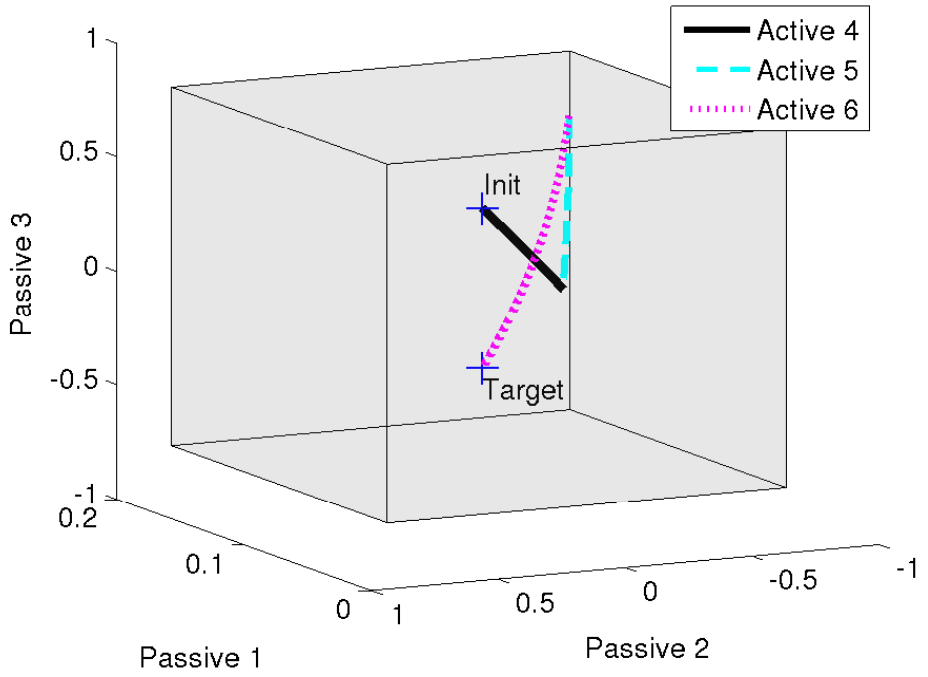


Figure 4.8: Optimal path of passive states ($m_{SW}(n_e) < \hat{m}_{a0} + 1$)

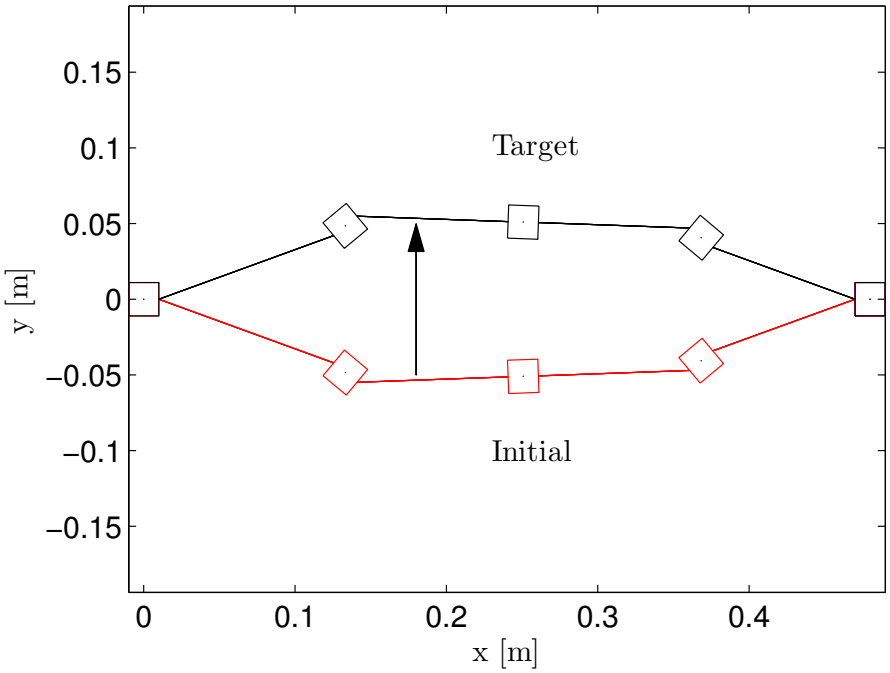


Figure 4.9: Initial and target configuration ($m_{SW}(n_e) = \hat{m}_{a0} + 1$)

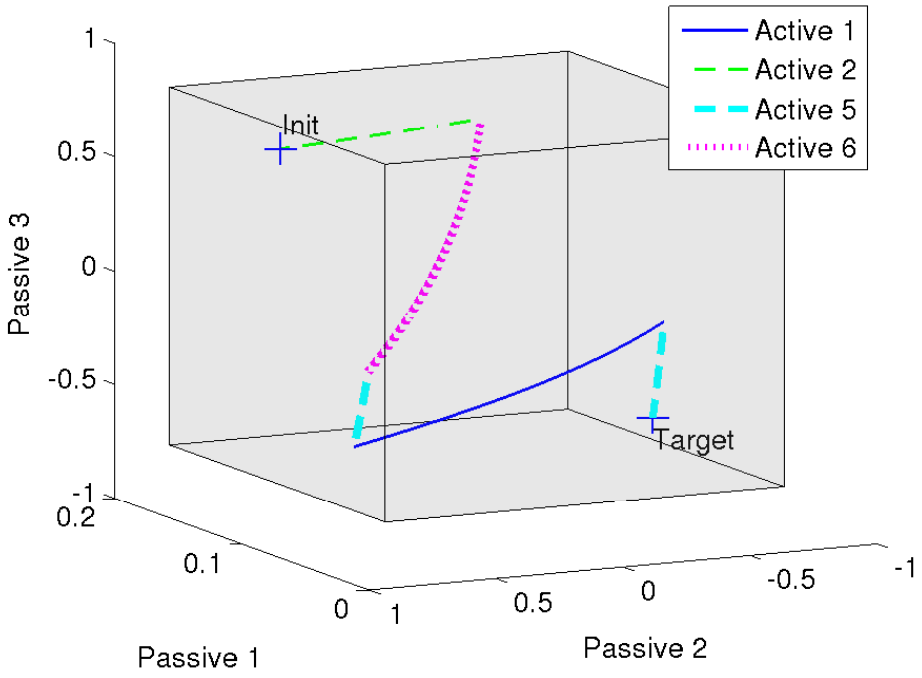


Figure 4.10: Optimal path of passive states ($m_{SW}(n_e) = \hat{m}_{a0} + 1$)

Suboptimal Solution: $m_{SW}(n_e) > \hat{m}_{a0} + 1$

Figure 4.11 shows another initial and target configuration of the mechanism. One can see that $\hat{m}_{a0} = 6$ active joints are not in the target state in the beginning. A solution with $m_{SW}(n_e) = 8$ switches was found ({3 4 5 2 1 6 1 6}). This means that a suboptimal solution was found and an optimal solution with 7 switches might exist. To find such a sequence, one has to discretize the state space, but the finding of an optimal solution cannot be guaranteed. The A*-algorithm generated 323 nodes (initial and target node included), until the solution was found. A run of a graph-search with a discretization of $n_d = 40$ using $m_{SW} = 8$ as lower bound did not find a solution. Thus, we assume that the solution found before is an optimal solution. The path of the passive states, using the sequence with 8 switches, is shown in Figure 4.12. One can see that no joint constraints are violated during the state transfer from $\boldsymbol{x}_0$ to $\boldsymbol{x}_e$.

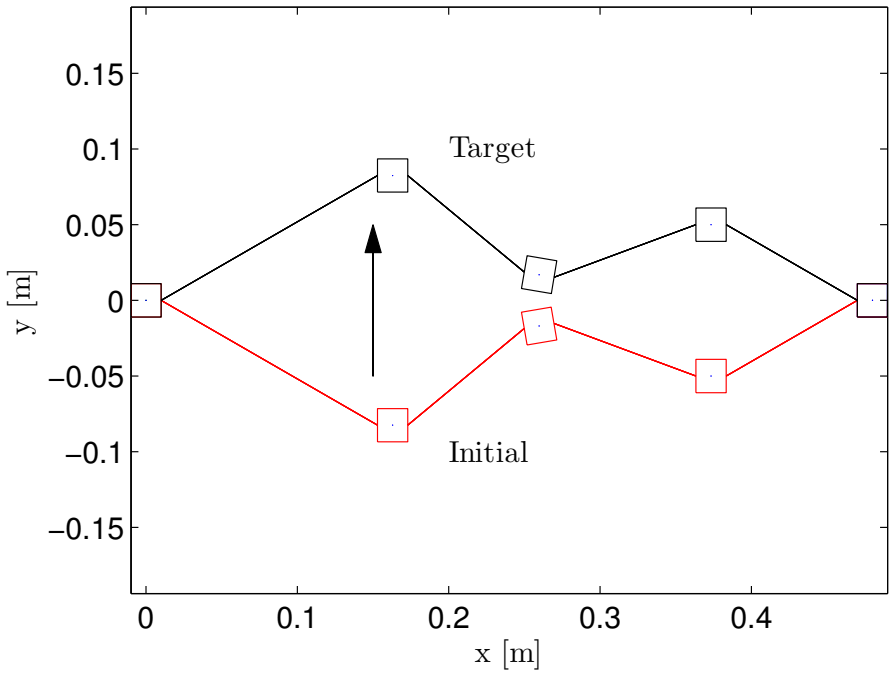


Figure 4.11: Initial and target configuration ($m_{SW}(n_e) > \hat{m}_{a0} + 1$)

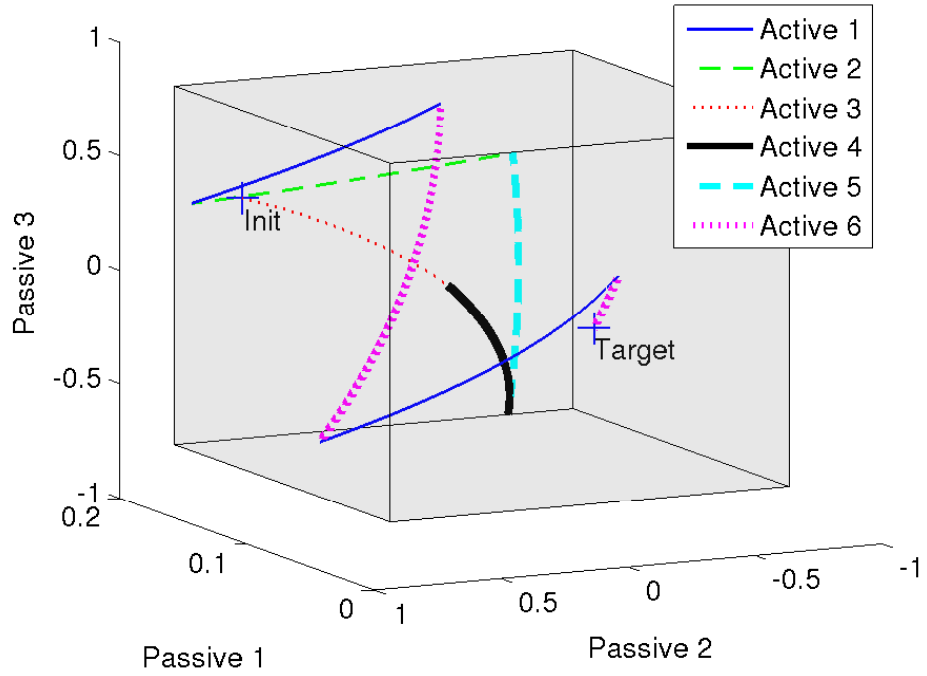


Figure 4.12: (Sub-)optimal path of passive states ($m_{SW}(n_e) > \hat{m}_{a0} + 1$)

4.5 Conclusion

In this chapter, the problem of finding an optimal switching sequence to transfer a parallel kinematics from an initial state into a target state was analyzed. Therefore, the problem was formulated as a hybrid optimization problem. The state model was formulated with the help of a hierarchical task space projection. After several simplifications the optimization problem could be simplified to a pure discrete optimization problem, and was solved with combinatorial graph search techniques. These simplifications are only allowed when assuming that the position control can follow the desired values of the active joints with no or only small errors. To deal with the cardinality of the combinatorial graph search, switching transitions were artificially introduced. With the help of these transitions (sub-)optimal solutions are found by searching an immensely reduced search tree. Statements, whether the found solution is optimal or sub-optimal, are given. In the case of a sub-optimal solution, a better (global optimal) solution might exist. Finding such a solution can, in some cases, be achieved by discretizing the state space and increasing the search tree. In most cases, the sub-optimal solution is sufficient.
In the future the method presented in this chapter must be extended to problems where more than one active joint can be actuated at a time. In this case, the cardinality of the graph increases as each combination of active joints, which can be actuated simultaneously, represents a separate path. The optimization problem is further complicated once the dynamics of the system is taken into account. In this case, the problem is not purely discrete any more, as the continuous part of the optimization is not constant.

5 Conclusions and Future Directions

5.1 Concluding Remarks

This thesis described the design and control of shape rendering interfaces (SRI), which can be used for haptic applications, where a high transparency in free space and full-hand interaction is desired.

The main goal of this thesis was the design of a SRI, with a sufficiently high resolution, which is easily extendable and which, in contrast to most state of the art interfaces, allows full bare hand interaction and is fully actuated. A further goal was to present control strategies for SRIs.

In Chapter 2 the design of a SRI, the Formable Object, was presented. The interface has 24 DoF and is designed as a parallel kinematics, which is fully actuated. After analyzing the determinacy and expandability of the device major requirements for the interface were defined. Basic elements, the so-called Telescopic Element and the Cardan Element, from which the FO is assembled, were presented. It was shown that both elements and not only one of the twos must be used in order to gain a high formability for the FO. It was further shown that not only the type of basic element, but also the way these basic elements are assembled, highly contributes to the formability of the FO. In order to find the optimal configuration out of a preselection of 9 configurations, we defined the so-called Formability Measure (FM), which was helpful to quantitatively compare different configurations in terms of formability. Based on this comparison the optimal configuration was selected and a prototype of the FO was designed. It was shown that construction-conditioned changes to the optimal design did not significantly change the formability of the mechanism. Details of the actuation, sensing and control principles were given. The prototype was evaluated in terms of maximum and minimum displayable curvatures, stiffness and dynamic performance. The FO can reach curvatures of 16.67 1/m (concave) and 11.36 1/m (convex) in the spherical case and 71.43 1/m (concave) and 15.15 1/m (convex) in the cylindrical case. A stiffness of > 4 N/mm was reached in normal direction to the interaction surface. A switch from one shape to another takes approximately 2 s.

Chapter 3 dealt with the kinematics of SRIs, especially with interfaces, which feature a parallel kinematics. After defining the problem of shape rendering, aspects, which have to be considered in the problem solution, were identified. The kinematics problem was solved in a differential way. The procedure of Nullspace projection outlined in [24], [25] or [26] was modified to the problem of shape rendering. Therefore, specific tasks namely the *Loop Constraint* task, the *Node Positioning* task and the *Shape Forming* task were introduced. For these tasks, which are executed on the Nullspace of the corresponding higher order tasks, task descriptors as well as the corresponding Jacobians, which link the task space to the joint

space of the mechanism, were defined. Two modes of shape rendering were distinguished: with or without user interaction. In the case when no user interaction was considered (stiff object without compliance) only the Loop Constraint task and the Shape Forming task had to be considered. If user interaction was considered all three tasks were used, because some nodes within the mechanism had to be controlled to exact positions to avoid unrealistic haptic impressions. Furthermore, different shape descriptions were discussed, namely implicit surfaces and polygon sets. It was shown that the framework introduced earlier can be used similarly for both shape descriptions. The chapter concluded with simulations of the different control modes, which showed that shapes can be rendered in real-time. The difference between rendering with and without user interaction was illustrated and it was shown that the procedure of shape rendering is independent of the shape description.

In Chapter 4, actions to reduce the needed amount of actuators were discussed. Using one actuator, which sequentially can be connected to the actuated joints of the mechanism with the help of clutches, was introduced as practical procedure to reduce the amount of actuators. The SRI was defined as a hybrid system, where each active joint defined a different system within the hybrid system description. Finding an optimal switching sequence to drive the kinematics from an initial configuration into a target configuration in minimum time was established by solving a hybrid control problem. Hereby a major challenge was to avoid joint limits of the passive joints of the mechanism. When assuming that the position control of the active joints can follow the desired values with no or only small errors, one can simplify the problem to a purely discrete problem, which can be solved with combinatorial graph search approaches. The approach was demonstrated on a planar parallel kinematics with 6DoF. The found solutions were discussed in terms of their optimality. Statements, if an optimal or a suboptimal solution is found could be made in most cases.

5.2 Outlook

This thesis provides new approaches to handle the challenges of designing and controlling SRIs. In the following, possible extensions of this work and collocated future research directions are pointed out.
As within shape rendering the forming of physical objects is requested, a highly perfected hardware is demanded for the task. The FO presented in chapter 2 already showed a well-engineered device but improvements can be done with respect to the following points. The FO is mainly built from ABS parts manufactured by a rapid-prototyper. This is done in order to have maximum scope in the design process. In a future version, these parts must be replaced by metallic parts (e.g. aluminum). This replacement will add strength to the mechanism, which, along with stronger actuators, will increase the stiffness of the device and make the mechanism long lasting. As can be seen from the results in chapter 2, precise position control is limited because high friction in the cable transmission leads to stick and slip effects. Reducing this friction by shortening the cables and choosing optimized ways to install the cables is an important task in future developments. Figures A.3 - A.6 illustrate how friction can be reduced by redesigning the motor module and using shorter cables lengths and better installations of the cables. Alternatively, other actuation principles can be con-

sidered. Hydraulic actuation represents a promising actuation principle because, in contrast to pneumatics, the fluid is incompressible and thus, a high stiffness can be gained for the device. Further, the nodes of the FO must be covered by an elastic layer, so that a continuous surface can be explored by the user. This layer must be stiff enough so that the regions between the nodes are not perceived too soft and elastic enough so that the mechanism is not limited in its motion. Another important task is the miniaturization, and in connection to this the extension of the FO. The prototype presented in this thesis has a node distance of about 40 mm, which is higher than state-of-the-art developments On the other hand, if one aims for rendering very detailed objects, this resolution will not be sufficient. The author believes that for miniaturizing the FO one needs to focus on two major aspects: Batch fabrication (e.g. MEMS-technology) with compliant joints [35], and integrated miniature actuation and sensing (e.g. electroactive polymers) [83].
If miniaturization is achievable, the amount of actuators increases drastically and thus, the approaches considered in chapter 4 become important. First of all, hardware modifications to the FO and its motor module need to be made in order to enable sequential actuation. As, when using discrete optimization more than one switching sequences may deliver the same value for the cost function, one might consider weighting these solutions in terms of errors to the target shape. This can be done by modifying the cost function and adding a cost term for the shape error. By doing so, one can identify a switching sequence which minimizes the shape error over the time interval of the deformation, while the time to reach the target state is minimal. A subject who touches the interface before the target position is reached will experience a smaller shape error compared to other sequences. Furthermore, sequences where specific joints are actuated simultaneously with one actuator must be considered. As mentioned before, we aim for a design where one actuator can be connected to one active joint and the other active joints are fixed. With a suitable hardware design, one can connect as many active joints to the actuator as are available in the mechanism. If multiple active joints must be driven in the same direction, these joints can be actuated simultaneously and the time until the target configuration is reached is reduced significantly by this measure.
From a kinematic point of view, one has to further consider different shape descriptions. In this thesis, we considered shapes given by either implicit functions or polygon data sets. The polygon data set is restricted to triangular polygons. This is a common description, but different polygons must be considered in future works. Furthermore, shape descriptions in the form of splines must be analyzed because such a description is common in many CAD-software tools.

A Appendix

A.1 Proof of Determinacy

A.1.1 Design 1

$$
\begin{aligned}
N_1(s_1) &= 4s_1^2 + 4s_1 + 1 \\
j_1(s_1) &= 6s_1^2 + 6s_1 \\
f_{i,1} &= 4 \\
n_1(s_1) &= N_1(s_1) - 1 \\
M_1(s_1) &= 12s_1^2 + 12s_1 \\
a_1(s_1) &= 2j_1 = 12s_1^2 + 12s_1 \\
M_1(s_1) &= a_1(s_1) \text{ q.e.d.}
\end{aligned} \tag{A.1}
$$

A.1.2 Design 2

$$
\begin{aligned}
N_2(s_2) &= 8s_2 + 1 \\
j_2(s_2) &= 12s_2 \\
f_{i,2} &= 4 \\
n_2(s_2) &= N_2(s_2) - 1 \\
M_2(s_2) &= 24s_2 \\
a_2(s_2) &= 2j_2 = 24s_2 \\
M_2(s_2) &= a_2(s_2) \text{ q.e.d.}
\end{aligned} \tag{A.2}
$$

A.1.3 Design 3

$$
\begin{aligned}
N_3(s_3) &= 9^{s_3} \\
j_3(s_3) &= \sum_{i=0}^{s-1} 9^i \\
f_{i,2} &= 4 \\
n_3(s_3) &= N_3(s3) - 1 \\
M_3(s_3) &= 6(9^{s_3} - 1) - 24 \sum_{i=0}^{s-1} 9^i = 48 \sum_{i=0}^{s-1} 9^i \\
\text{with } 9^{s_3} - 1 &= 8 \sum_{i=0}^{s-1} 9^i \text{ (Geometric Sum)} \\
a_3(s_3) &= 2j_3 = 24s_3 \\
M_3(s_3) &= a_3(s_3) \text{ q.e.d.}
\end{aligned} \tag{A.3}
$$

A.2 Details on FO design

Figure A.1 and A.2 show the exploded view of the Telescopic Element and the Cardan Element, respectively.

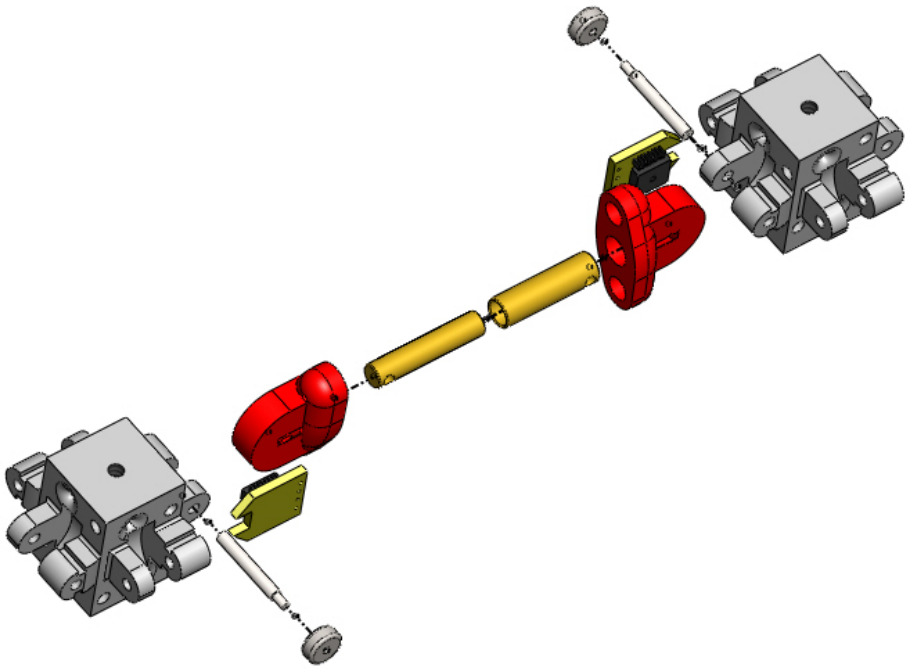


Figure A.1: Exploded view of the Telescopic Element

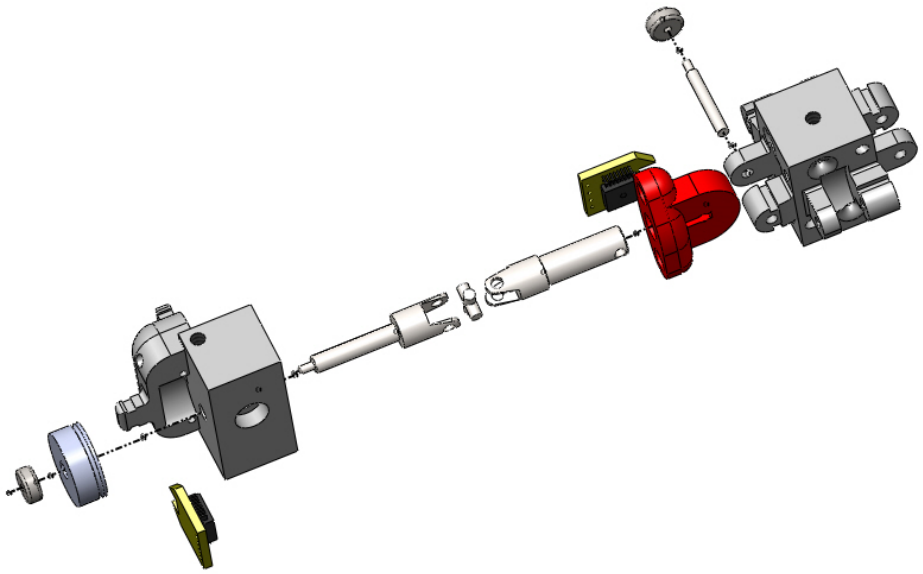


Figure A.2: Exploded view of the Cardan Element

A.3 Revised Design of the FO and the Motor Module

A revised version of the FO and its motor module is outlined in the following figures. Figure A.3 and A.4 show the FO with the same dimensions and configuration of DoF as described in chapter 2. The difference is that the cable transmission is arranged such that the cables connect to the corresponding joints from below. In Figure A.4 one can see the supplies for the cables. Corresponding modifications must be made to the motor module as well, such that the cables stick out to the top (cf. Figure A.5). By doing so the cable length and thus, the friction in the cables can be minimized. The assembly of FO and motor module is shown in Figure A.6. The revised Telescopic and Cardan Element is shown in Figure A.7 and A.8, while the corresponding exploded views are shown in Figure A.9 and A.10

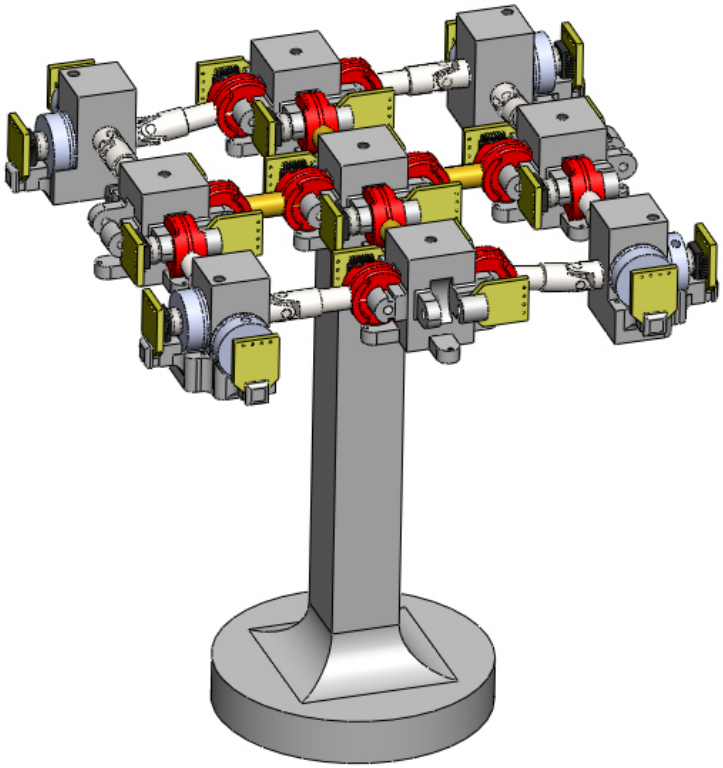


Figure A.3: FO with actuation supply from below

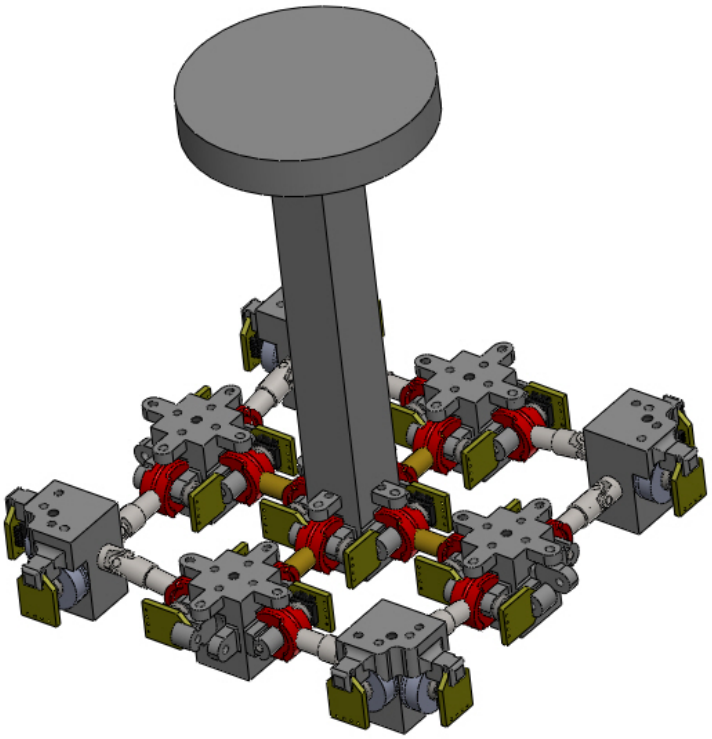


Figure A.4: Bottom view of revised FO

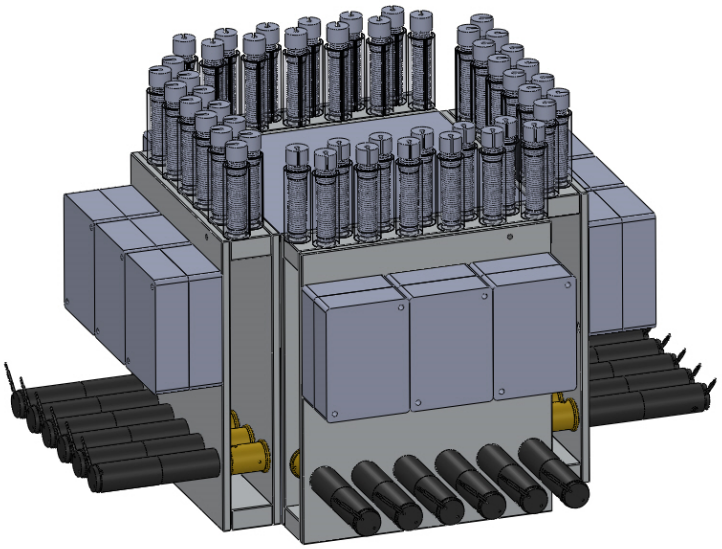


Figure A.5: Revised motor module with actuation supply at top

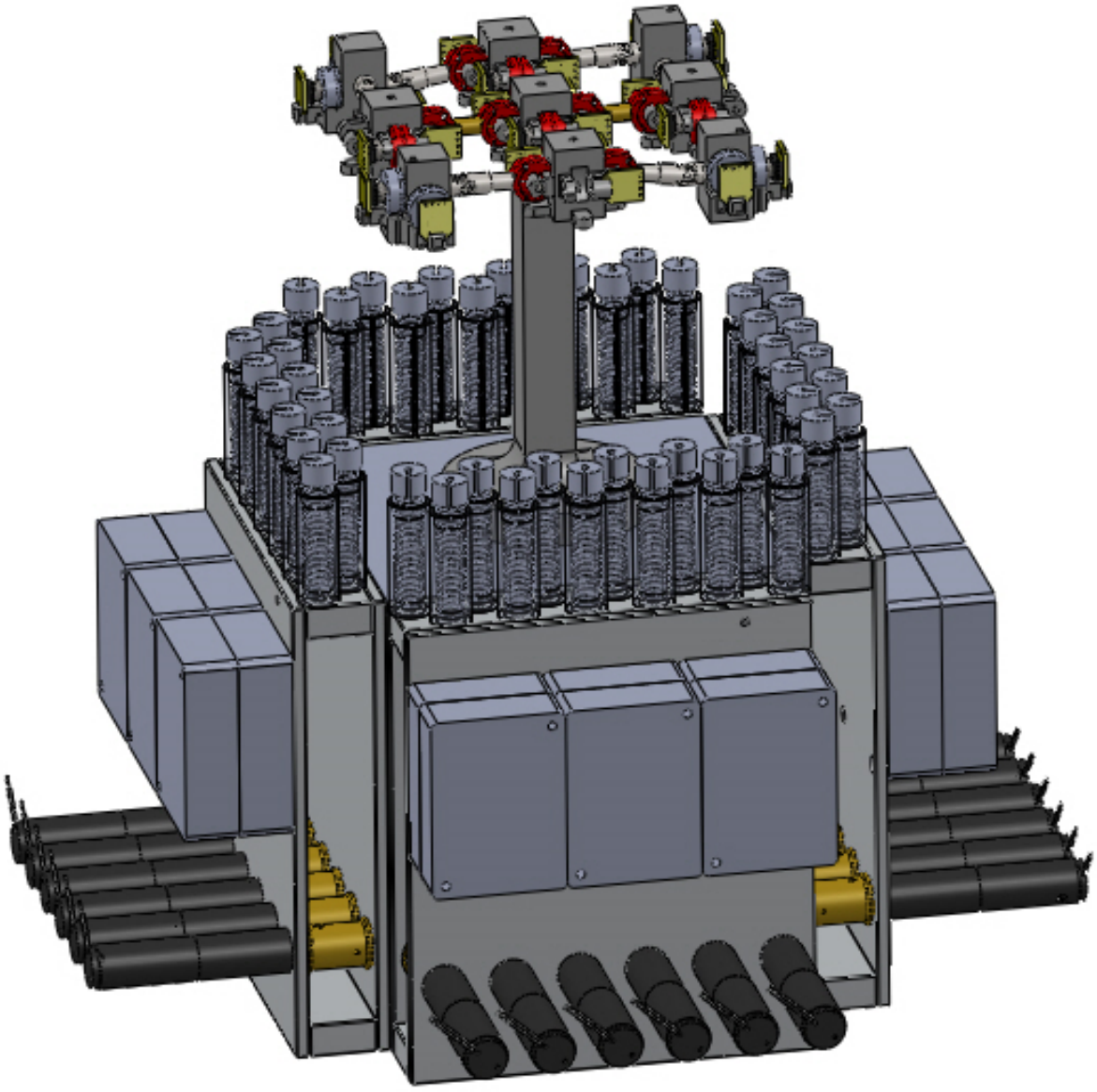


Figure A.6: Assembly of revised FO and motor module

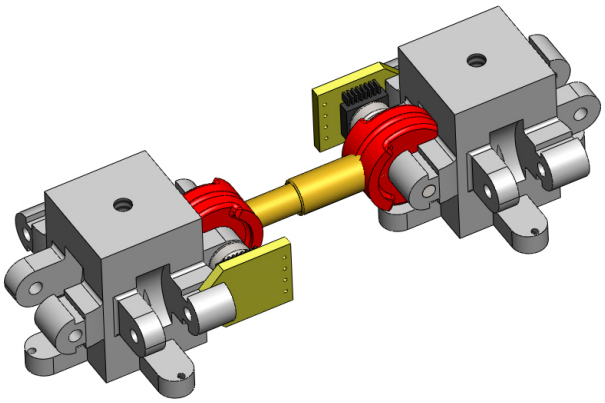


Figure A.7: Revised Telescopic Element

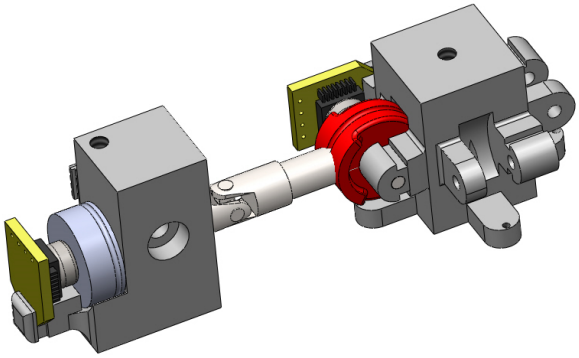


Figure A.8: Revised Cardan Element

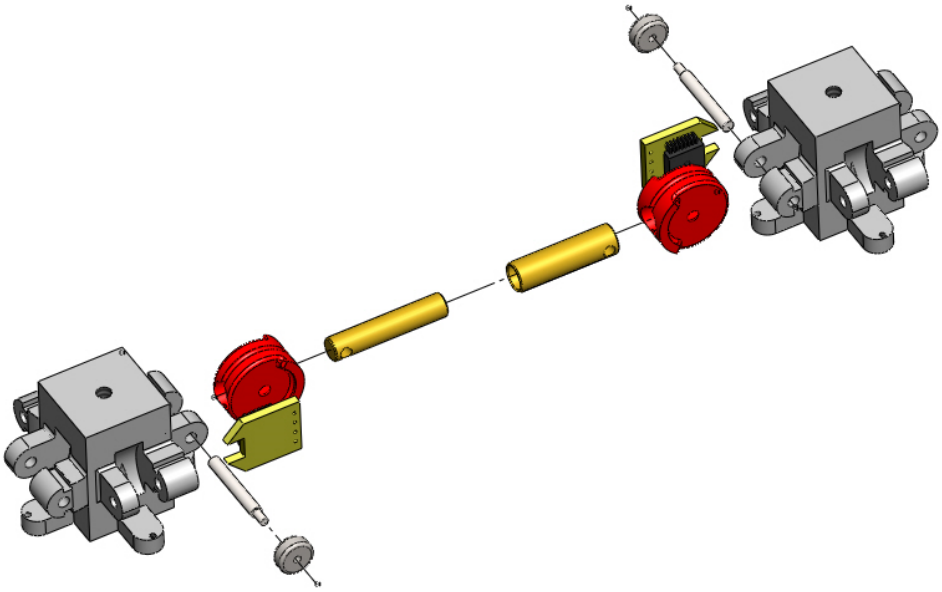


Figure A.9: Exploded view of revised Telescopic Element

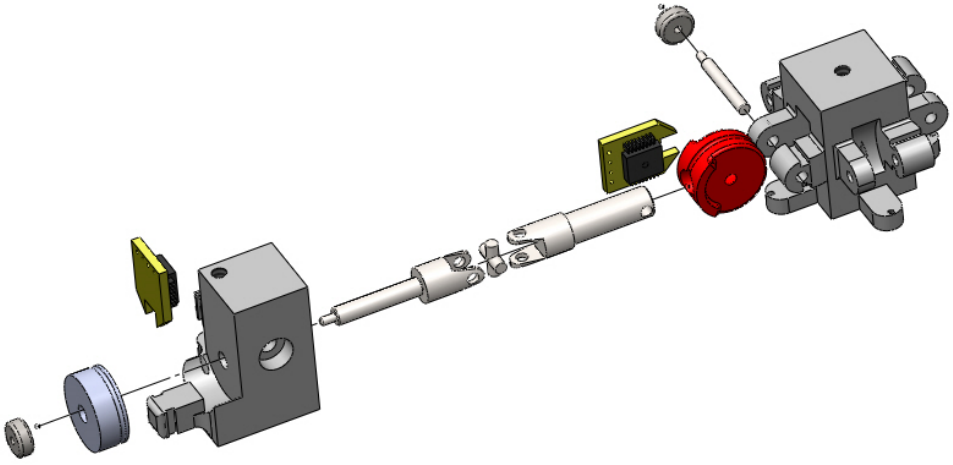


Figure A.10: Exploded view of revised Cardan Element

Bibliography

[1] K.-U. Kyung, S.-C. Kim, D.-S. Kwon, and M. A. Srinivasan, "Texture display mouse kat: Vibrotactile pattern and roughness display," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 478–483, IEEE, 2006.

[2] H. Kajimoto, N. Kawakami, S. Tachi, and M. Inami, "Smarttouch: Electric skin to touch the untouchable," *Computer Graphics and Applications, IEEE*, vol. 24, no. 1, pp. 36–43, 2004.

[3] Y. Yokokohji, J. Kinoshita, and T. Yoshikawa, "Path planning for encountered-type haptic devices that render multiple objects in 3d space," in *Virtual Reality, 2001. Proceedings. IEEE*, pp. 271–278, IEEE, 2001.

[4] T. Hoshi, T. Iwamoto, and H. Shinoda, "Non-contact tactile sensation synthesized by ultrasound transducers," in *EuroHaptics conference, 2009 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2009. Third Joint*, pp. 256–260, IEEE, 2009.

[5] D. A. Lawrence, "Stability and transparency in bilateral teleoperation," *Robotics and Automation, IEEE Transactions on*, vol. 9, no. 5, pp. 624–637, 1993.

[6] G. J. Raju, G. C. Verghese, and T. B. Sheridan, "Design issues in 2-port network models of bilateral remote manipulation," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pp. 1316–1321, IEEE, 1989.

[7] C. Passenberg, A. Peer, and M. Buss, "A survey of environment-, operator-, and task-adapted controllers for teleoperation systems," *Mechatronics*, vol. 20, no. 7, pp. 787–801, 2010.

[8] T. B. Sheridan, "Telerobotics," *Automatica*, vol. 25, no. 4, pp. 487–507, 1989.

[9] M. Benali-Khoudja, M. Hafez, J.-M. Alexandre, and A. Kheddar, "Tactile interfaces: a state-of-the-art survey," in *Int. Symposium on Robotics*, vol. 31, 2004.

[10] V. Hayward, O. R. Astley, M. Cruz-Hernandez, D. Grant, and G. Robles-De-La-Torre, "Haptic interfaces and devices," *Sensor Review*, vol. 24, no. 1, pp. 16–29, 2004.

[11] A. Peer, Y. Komoguchi, and M. Buss, "Towards a mobile haptic interface for bimanual manipulations," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 384–391, IEEE, 2007.

[12] M. Ueberle and M. Buss, "Design, control, and evaluation of a new 6 dof haptic device," in *In Proc. of the 2002 IEEE/RSJ Int. Conf. on Intellig. Rob. and Syst*, pp. 2949–2954, 2002.

[13] P. Weangsima, K. Fujita, and T. Honda, "A study of haptic representation of virtual plain wall," in *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*, pp. 323–327, IEEE, 2004.

[14] K. Hirota and M. Hirose, "Development of surface display," in *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*, pp. 256–262, IEEE, 1993.

[15] S. Nakagawara, H. Kajimoto, N. Kawakami, S. Tachi, and I. Kawabuchi, "An encounter-type multi-fingered master hand using circuitous joints," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 2667–2672, IEEE, 2005.

[16] T. Nojima, M. Inami, T. Maeda, and S. Tachi, "Applying an'encounter-type'haptic display to telexistence," in *Proceedings of the Virtual Reality Society of Japan Annual Conference*, vol. 4, pp. 395–398, 1999.

[17] C. Heinze, "Modelling intention recognition for intelligent agent systems," tech. rep., DTIC Document, 2004.

[18] M. E. Bratman, D. J. Israel, and M. E. Pollack, "Plans and resource-bounded practical reasoning," *Computational intelligence*, vol. 4, no. 3, pp. 349–355, 1988.

[19] H. A. Kautz and J. F. Allen, "Generalized plan recognition.," in *AAAI*, vol. 86, pp. 32–37, 1986.

[20] M. Wooldridge, "Agent-based software engineering," *IEE Proceedings-software*, vol. 144, no. 1, pp. 26–37, 1997.

[21] H. Iwata, H. Yano, F. Nakaizumi, and R. Kawamura, "Project FEELEX: adding haptic surface to graphics," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, p. 476, ACM, 2001.

[22] D. Rosen, A. Nguyen, and H. Wang, "On the Geometry of Low Degree-of-Freedom Digital Clay Human-Computer Interface Devices," in *Proceedings ASME Computers and Information in Engineering Conference, Chicago, Sept*, pp. 2–6, 2003.

[23] V. Hayward and O. R. Astley, "Performance measures for haptic interfaces," in *Robotics Research*, pp. 195–206, Springer, 2000.

[24] B. Dariush, M. Gienger, A. Arumbakkam, Y. Zhu, B. Jian, K. Fujimura, and C. Goerick, "Online transfer of human motion to humanoids," *International Journal Of Hr: Humanoid Robotics*, vol. 6, no. 2, p. 265, 2009.

[25] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2, no. 4, pp. 505–518, 2005.

[26] W. Decré, R. Smits, H. Bruyninckx, and J. De Schutter, "Extending itasc to support inequality constraints and non-instantaneous task specification," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 964–971, IEEE, 2009.

[27] E. Ruffaldi, C. A. Avizzano, P. Tripicchio, A. Frisoli, and M. Bergamasco, "Surface perception in a large workspace encounter interface," in *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pp. 21–26, IEEE, 2008.

[28] Y. Yokokohji, R. L. Hollis, and T. Kanade, "What you can see is what you can feel-development of a visual/haptic interface to virtual environment," in *Virtual Reality Annual International Symposium, 1996., Proceedings of the IEEE 1996*, pp. 46–53, IEEE, 1996.

[29] Y. Yokokohji, N. Muramori, Y. Sato, T. Kikura, and T. Yoshikawa, "Design and path planning of an encountered-type haptic display for multiple fingertip contacts based on the observation of human grasping behavior," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 2, pp. 1986–1991, IEEE, 2004.

[30] Y. Yokokohji, N. Muramori, Y. Sato, and T. Yoshikawa, "Designing an encountered-type haptic display for multiple fingertip contacts based on the observation of human grasping behaviors," *The International Journal of Robotics Research*, vol. 24, no. 9, pp. 717–729, 2005.

[31] S. Goldstein, J. Campbell, and T. Mowry, "Programmable matter," *Computer*, vol. 38, no. 6, pp. 99–101, 2005.

[32] V. Zykov, A. Chan, and H. Lipson, "Molecubes: An open-source modular robotics kit," in *Proc. IROS*, vol. 7, 2007.

[33] K. Hirota and M. Hirose, "Surface display: Concept and implementation approaches," in *Proceedings of the Fifth International Conference on Artificial Reality and Tele-Existence*, pp. 185–192, 1995.

[34] P. Bosscher and I. Ebert-Uphoff, "Digital clay: Architecture designs for shape-generating mechanisms," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 1, pp. 834–841, IEEE, 2003.

[35] P. Bosscher and I. Ebert-Uphoff, "A novel mechanism for implementing multiple collocated spherical joints," in *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 336–341, Citeseer, 2003.

[36] M. Bordegoni, U. Cugini, M. Covarrubias, and M. Antolini, "A Force and Touch Sensitive Self-deformable Haptic Strip for Exploration and Deformation of Digital Surfaces," *Haptics: Generating and Perceiving Tangible Sensations*, pp. 65–72, 2010.

[37] A. Mazzone, *Deformable mechanical structure for physical generation of objects and provision of wide area haptic feedback*. VDI-Verl., 2005.

[38] A. Mazzone, C. Spagno, and A. Kunz, "A haptic feedback device based on an active mesh," in *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 188–195, ACM, 2003.

[39] S. Klare, D. Forssilow, and A. Peer, "Formable object: A new haptic interface for shape rendering," in *Worldhaptics (WH), 2013 IEEE International Conference on*.

[40] A. Swanson, I. Matev, and G. De Groot, "The strength of the hand," *Bulletin of prosthetics research*, vol. 10, no. 14, pp. 145–153, 1970.

[41] S. Klare and A. Peer, "Inverse kinematics for shape rendering interfaces," in *Robotics and Automation, 2013. Proceedings. ICRA'13. IEEE International Conference on*.

[42] Z. Huang, Q. Li, and H. Ding, *Theory of parallel mechanisms*, vol. 6. Springer, 2013.

[43] R. Byrd, M. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, 1999.

[44] J. Merlet, *Parallel robots*. Springer-Verlag New York Inc, 2006.

[45] R. Clavel and C. W. Burckhardt, "Conception d'un robot parallèle rapide à 4 degrés de liberté," tech. rep., 1991.

[46] D. Stewart, "A platform with six degrees of freedom," *Proceedings of the institution of mechanical engineers*, vol. 180, no. 1, pp. 371–386, 1965.

[47] A. Mazzone and A. Kunz, "Sketching the future of the smartmesh wide area haptic feedback device by introducing the controlling concept for such a deformable multi-loop mechanism," *Links*, vol. 3, p. 248, 2005.

[48] J. E. Colgate and J. M. Brown, "Factors affecting the z-width of a haptic display," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 3205–3210, IEEE, 1994.

[49] Y. Chen and I. Walker, "A consistent null-space based approach to inverse kinematics of redundant robots," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pp. 374–381, IEEE, 1993.

[50] J. Luh, M. Walker, and R. Paul, "Resolved-acceleration control of mechanical manipulators," *Automatic Control, IEEE Transactions on*, vol. 25, no. 3, pp. 468–474, 1980.

[51] S. A. Cover, N. F. Ezquerra, J. F. O'Brien, R. Rowe, T. Gadacz, and E. Palm, "Interactively deformable models for surgery simulation," *Computer Graphics and Applications, IEEE*, vol. 13, no. 6, pp. 68–75, 1993.

[52] A. Joukhadar and C. Laugier, "Fast dynamic simulation of rigid and deformable objects," in *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 2, pp. 305–310, IEEE, 1995.

[53] M. Bro-Nielsen and S. Cotin, "Real-time volumetric deformable models for surgery simulation using finite elements and condensation," in *Computer graphics forum*, vol. 15, pp. 57–66, Wiley Online Library, 1996.

[54] W. M. Hsu, J. F. Hughes, and H. Kaufman, "Direct manipulation of free-form deformations," in *ACM Siggraph Computer Graphics*, vol. 26, pp. 177–184, ACM, 1992.

[55] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in *ACM Siggraph Computer Graphics*, vol. 20, pp. 151–160, ACM, 1986.

[56] H. Hoffman, D. Stredney, and S. Weghorst, "Force interactions in laparoscopic simulations: Haptic rendering of soft tissues," *Medicine Meets Virtual Reality: Art, Science, Technology: Healthcare (R) Evolution*, vol. 50, p. 385, 1998.

[57] H. Çakmak and U. Kühnapfel, "Animation and simulation techniques for vr-training systems in endoscopic surgery," in *Computer Animation and Simulation 2000*, pp. 173–185, Springer, 2000.

[58] A. A. Stanley, J. C. Gwilliam, and A. M. Okamura, "Haptic jamming: A deformable geometry, variable stiffness tactile display using pneumatics and particle jamming," in *World Haptics Conference (WHC), 2013*, pp. 25–30, IEEE, 2013.

[59] K. Salisbury and C. Tarr, "Haptic rendering of surfaces defined by implicit functions," in *ASME Dynamic Systems and Control Division*, vol. 61, pp. 61–67, 1997.

[60] C. B. Zilles and J. K. Salisbury, "A constraint-based god-object method for haptic display," in *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 3, pp. 146–151, IEEE, 1995.

[61] C.-H. Ho, C. Basdogan, and M. A. Srinivasan, "Efficient point-based rendering techniques for haptic display of virtual objects," *Presence: Teleoperators and Virtual Environments*, vol. 8, no. 5, pp. 477–491, 1999.

[62] C. Häger-Ross and M. H. Schieber, "Quantifying the independence of human finger movements: comparisons of digits, hands, and movement frequencies," *The Journal of Neuroscience*, vol. 20, no. 22, pp. 8542–8550, 2000.

[63] L. S. Pontrëiìagin, *The mathematical theory of optimal processes*, vol. 4. CRC Press, 1962.

[64] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control," in *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, vol. 4, pp. 4228–4234, IEEE, 1994.

[65] H. J. Sussmann, "A maximum principle for hybrid optimal control problems," in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, vol. 1, pp. 425–430, Ieee, 1999.

[66] M. Broucke, M. D. Di Benedetto, S. Di Gennaro, and A. Sangiovanni-Vincentelli, "Optimal control using bisimulations: Implementation," in *Hybrid Systems: Computation and Control*, pp. 175–188, Springer, 2001.

[67] M. S. Shaikh and P. E. Caines, "On the optimal control of hybrid systems: Optimization of trajectories, switching times, and location schedules," in *Hybrid systems: Computation and control*, pp. 466–481, Springer, 2003.

[68] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[69] S. Hedlund and A. Rantzer, "Optimal control of hybrid systems," in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, vol. 4, pp. 3972–3977, IEEE, 1999.

[70] B. De Schutter, "Optimal control of a class of linear hybrid systems with saturation," *SIAM Journal on Control and Optimization*, vol. 39, no. 3, pp. 835–851, 2000.

[71] B. Lincoln and A. Rantzer, "Optimizing linear system switching," in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, vol. 3, pp. 2063–2068, IEEE, 2001.

[72] A. Bemporad, A. Giua, and C. Seatzu, "A master-slave algorithm for the optimal control of continuous-time switched affine systems," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 2, pp. 1976–1981, IEEE, 2002.

[73] X. Xu and P. J. Antsaklis, "Quadratic optimal control problems for hybrid linear autonomous systems with state jumps," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 4, pp. 3393–3398, IEEE, 2003.

[74] P. E. Caines and M. S. Shaikh, "Optimality zone algorithms for hybrid systems: Efficient algorithms for optimal location and control computation," in *Hybrid Systems: Computation and Control*, pp. 123–137, Springer, 2006.

[75] M. S. Shaikh and P. E. Caines, "On the hybrid optimal control problem: theory and algorithms," *Automatic Control, IEEE Transactions on*, vol. 52, no. 9, pp. 1587–1603, 2007.

[76] O. Stursberg, "A graph search algorithm for optimal control of hybrid systems," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 2, pp. 1412–1417, IEEE, 2004.

[77] O. Stursberg, "Dynamic optimization of processing systems with mixed degrees of freedom," in *Dynamics and Control of Process Systems 2004 (DYCOPS-7): A Proceedings Volume from the 7th IFAC Symposium, Cambridge, Massachusetts, USA, 5-7 July 2004*, p. 649, Access Online via Elsevier, 2004.

[78] B. Passenberg and O. Stursberg, "Graph search for optimizing the discrete location sequence in hybrid optimal control," in *3rd IFAC Conf. on Analysis and Design of Hybrid Systems*, pp. 304–309, 2009.

[79] N. R. Sturtevant, A. Felner, M. Likhachev, and W. Ruml, "Heuristic search comes of age.," in *AAAI*, 2012.

[80] M. Buss, "Methoden zur regelung hybrider dynamischer systeme," *VDI Fortschritt-Bericht, Reihe*, vol. 8, 2002.

[81] M. Papageorgiou, *Optimierung: statische, dynamische, stochastische Verfahren*. Springer DE, 2012.

[82] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[83] Y. Bar-Cohen, "Electroactive polymers: current capabilities and challenges," in *SPIE's 9th Annual International Symposium on Smart Structures and Materials*, pp. 1–7, International Society for Optics and Photonics, 2002.