Reaktive Kollisionsvermeidungsstrategien für Roboter in der direkten Mensch-Roboter Interaktion

Diplomarbeit

Rico Belder

DLR

Institut für Robotik und Mechatronik	BJ.: 2010
	IB.Nr.: 515-10-11

DIPLOMARBEIT

REAKTIVE KOLLISIONSVERMEIDUNGSSTRATEGIEN FÜR ROBOTER IN DER DIREKTEN MENSCH-ROBOTER INTERAKTION

Freigabe:	Der Bearbeiter:	Unterschriften
	Rico Belder	
	Betreuer:	
	Dipl. Ing., MSc. Sami Haddadin	
	Der Institutsdirektor	
	Prof. Dr. G. Hirzinger	

Dieser Bericht enthält 112 Seiten, 63 Abbildungen und 9 Tabellen

Ort: Oberpfaffenhofen	Datum: 30.09.2010	Bearbeiter: Rico Belder	Zeichen:

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Elektrotechnik und Informationstechnik

Institut für Automatisierungstechnik

Aufgabenstellung für die Diplomarbeit

für

Herrn Rico Belder

Reaktive Kollisionsvermeidungsstrategien für Roboter in der direkten Mensch-Roboter Interaktion

Hintergrund:
Künftige Robotergenerationen werden im industriellen Umfeld eng mit dem Menschen zusammenarbeiten. Hierzu sind insbesondere ein sicheres Roboter-Design, Methoden zur nachgiebigen Regelung und Kollisionserkennung mit geeigneten Strategien nötig. Darüber hinaus ist es von großer Bedeutung flexible Methoden zur Bewegungsgenerierung zur Verfügung zu stellen, die das potentiell komplexe statische Umfeld des Roboters berücksichtigen und gleichzeitig sehr schnell auf sich dynamisch verändernde Umgebungsbedingungen reagieren.

Zielsetzung:
In der Diplomarbeit werden vorhandene reaktive Bewegungsalgorithmen mit Hinblick auf ihre Anwendbarkeit auf anthropomorphe Roboterarme untersucht. Ihre inhärenten Eigenschaften und Leistungsmerkmale gilt es insbesondere mit Hinblick auf ihre Nutzbarkeit für Anwendungen der physikalischen Mensch-Roboter Interaktion (pHRI) zu vergleichen. Hierzu sollen ausgewählte Methoden in 2D und 3D anhand von Beispielen steigender Komplexität simuliert und im Anschluss in eine vollständige dynamische Simulation des DLR Leichtbauroboter III (LBR-III) integriert werden. Zusätzlich zur Analyse der bestehenden Methoden sollen notwendige Erweiterungen und neue Ansätze, welche die besonderen Anforderungen der pHRI berücksichtigen, entwickelt werden. Die angewendeten sowie entworfenen Algorithmen sollen schließlich am realen Roboter im DLR Co-Worker-Szenario umgesetzt und validiert werden.

Folgende Arbeitsschritte sind durchzuführen:
- Literaturrecherche, Anforderungsdefinition, Erarbeitung eines Pflichtenheftes
- Simulation bestehender Ansätze für den LBR-III
- Bewertung der ausgewählten Systeme zur Echtzeit-Optimalität und der sonstigen gewählten Kriterien
- Erweiterung der gewählten Methoden in Bezug auf die Anforderungen der sicheren Mensch-Roboter-Interaktion
- Umsetzung für den LBR-III und Integration in bestehende Anwendungen
- Test und quantitative Leistungsanalyse der implementierten Lösung

Betreuer:	Dipl.Ing., MSc. Sami Haddadin (DLR)
Ausgehändigt am:	15.02.2010
Einzureichen am:	13.08.2010

Prof. Dr.-Ing. B. Bäker
Vorsitzender des PA Mechatronik

TU Dresden Prüfungsamt Fakultät Elektrotechnik und Informationstechnik

Prof. Dr. techn. K. Janschek
Verantw. Hochschullehrer

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Elektrotechnik und Informationstechnik

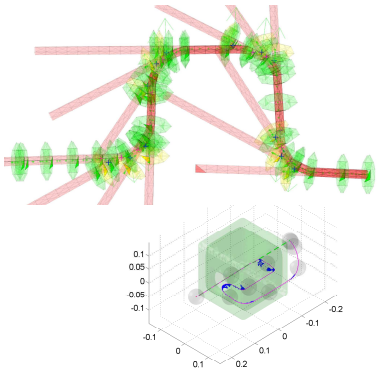
Institut für Automatisierungstechnik

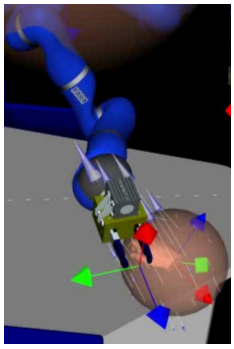
Reaktive Kollisionsvermeidungsstrategien für Roboter in der direkten Mensch-Roboter Interaktion

Kurzfassung

Künftige Robotergenerationen werden im privaten als auch im industriellen Bereich z.B. als sogenannte Co-Worker eng mit dem Menschen zusammenarbeiten. Da die Arbeitsbereiche von Robotern sehr variable sein können, müssen diese in der Lage sein in einer dynamischen und teilweise unbekannten Umgebung zu agieren. Hierfür müssen reaktive Bewegungen generiert, Kollisionen vermieden und für erwünschten oder unvermeidlichen Kontakt robuste und sichere Reaktionsstrategien entwickelt werden.

Zu diesen Zweck wurden 2D, 3D und 6D Simulationsumgebungen steigender Komplexität in umfangreichen Testszenarien entwickelt. Ausgewählte Algorithmen wurden in komplexen 2D-Simulationen in Bezug auf ihre Fähigkeiten zur Lösung der genannten Probleme verglichen und nach erneuter Selektion zu einer hybriden Strategie erweitert. Abschließend wurde eine 6D MATLAB/Simulink/Stateflow Implementierung eines Circular-Potential Fields Ansatzes verwendet, um systematische Analysen anhand unterschiedlicher Testfelder und Szenarien mit sich dynamisch bewegenden Menschen durchzuführen. Darüber hinaus wurde ein leistungsfähiger Algorithmus zur taktilen Exploration komplexer ebener 3D Drahtelemente mit vorab unbekannter Struktur entwickelt und getestet.





Betreuer:	Dipl. Ing., MSc. Sami Haddadin (DLR)
Hochschullehrer:	Prof. Dr. techn. Klaus Janschek
Tag der Einreichung:	10.09.2010

DIPLOMARBEIT

Bearbeiter: Rico Belder

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Elektrotechnik und Informationstechnik

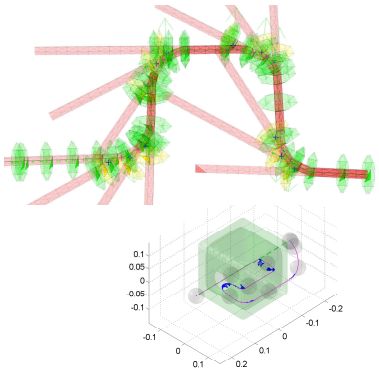
Institut für Automatisierungstechnik

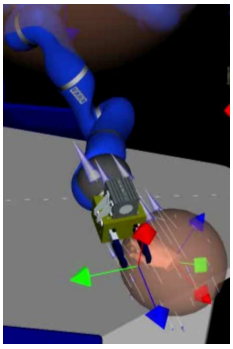
Reactive Collision Avoidance Strategies for Robots in direct Human-Robot Interaction

Abstract

In the near future robots are sought to become an integral part of human everyday life. Also in industrial settings robotic Co-Workers are expected to become a commodity. Even though the particular application areas may vastly change, a robot always needs to act in a dynamic and partially unknown environment. It shall reactively generate motions and prevent upcoming collisions. If contact is desired or inevitable, it has to handle it robustly and safely.

In order to evaluate existing algorithms an extensive simulation environment with test scenarios of rising complexity in 2D, 3D, and 6D was developed. After an initial analysis in rather complex 2D simulations, particularly well suited ones were extended to 3D as well as 6D, and combined into a hybrid strategy. Finally, the 6D MATLAB/Simulink/StateFlow implementation of a hybrid Circular & Potential Fields approach is used to perform the experimental analysis for static multi-object parcours and to avoid dynamically moving humans in a 6D task motion. Furthermore, we developed and tested a high-performance algorithm for tactile exploration of complex planar 3D wire elements, whose structure is a-priori unknown.





Tutor:	Dipl. Ing., MSc. Sami Haddadin (DLR)
Supervisor:	Prof. Dr. techn. Klaus Janschek
Day of Submission:	10.09.2010

DIPLOMA THESIS

Author: Rico Belder

Danksagung

Diese Arbeit widme ich meinen Eltern, die mich schon immer bedingungslos auf meinem Lebensweg unterstützt und mir auch mein Studium, diese Diplomarbeit und damit so vielseitige, schöne und lehrreiche Erfahrungen ermöglichten.

Besonders möchte ich mich an dieser Stelle bei allen bedanken, die mich während der Erstellung dieser Arbeit unterstützt haben und ohne die sie nicht hätte zustande kommen können. Vielen Dank an Prof. Dr. techn. Klaus Janschek, der diese externe Diplomarbeit von Seiten der Technischen Universität Dresden betreut hat. Weiterhin möchte ich mich bei Prof. Dr.-Ing. Gerd Hirzinger und insbesondere bei Dr.-Ing. Alin Albu-Schäffer auf Seiten des Deutschen Zentrums für Luft und Raumfahrt für die Möglichkeit bedanken beim DLR meine Diplomarbeit durchführen zu können. Ganz besonders, über sein Vertrauen hinweg, möchte ich mich bedanken bei meinen Betreuer Sami Haddadin für seine Zeit, die er mit mir in Diskussion, beim Experimentieren mit den Robotern, oder beim Korrekturlesen der Arbeit verbracht hat und damit diese Diplomarbeit zu einer so lehrreichen Erfahrung für mich machte.

Natürlich möchte ich mich auch bei allen anderen Mitarbeitern des Instituts für Robotik und Mechatronik bedanken, die mir während meiner Diplomarbeit mit Rat und Tat zur Seite standen.

Abschließend möchte ich mich bei meinen Freunden aus Dresden, als auch aus München dafür bedanken, dass Sie diese Zeit des Ausnahmezustandes über sich ergehen ließen.

Contents

List of Figures

List of Tables

Symbols and Abbreviations

AO	Avoiding Object
CCD	Charge Coupled (Area Imaging) Device
CFs	Circular Fields
CPU	Central Processing Unit
DoF	Degrees of Freedom
FPGA	Field-Programmable Gate Array
GFLOPS	Giga Floating Point Operation Per Second
GPS	Global Positioning System
HNPFs	Human Navigation Potential Fields
HPFs	Harmonic Potential Fields
HRI	Human-Robot Interaction
LWR-III	DLR-Lightweight Robot III
OO	Obstacle Object
PFs	Potential Fields
pHRI	physical Human-Robot Interaction
RCA	Reactive Collision Avoidance or the functionality "generate Collision Avoidance Values"
RCU	Robot Control Unit
RRTs	Rapidly Exploring Random Trees
TCU	Task Control Unit
UAV	Unmanned Aerial Vehicle

1 Introduction

In the near future robots are sought to become an integral part of our daily life as multi-purpose service assistants in our homes. Apart from such domestic applications, flexible and versatile robots may also relieve us from monotonous and physically demanding work in industrial settings. In dangerous or even life-threatening surroundings, as e.g. deep-space or underwater, they may even replace humans entirely. Especially when being used in disaster areas or underground scenarios a robot may search and rescue people from hardly accessible locations. However, even though the particular application areas may vastly change, a robot needs to be able to act in a dynamic and partially unknown environment. It shall reactively generate motions and prevent upcoming collisions and if contact is desired or inevitable, it needs to robustly and safety handle it.

Especially physical Human-Robot Interaction (pHRI) is a field in which such behavior is of immanent importance. As human and robot shall collaborate very closely (see Fig. 1.1) the problem of generating "human-friendly" motions is of large interest. Even though the close interaction of human and robot in the domestic and industrial sector was always proclaimed to open up entirely new possibilities for service applications and production processes, several problems are still to be tackled before finally achieving this ambitious goal. A particularly important problem indeed is the generation of safe motions in human vicinity. However, up to now reactive motion capabilities were usually developed for mobile robots, where the complexity of the avoidance problem is limited by nature. For multi degree of freedom (DoF) articulated manipulators, on the other hand, only few algorithms that are tailored to their needs were developed. This is mainly due to the fact that such abilities were unnecessary in real-world applications. Articulated robots were exclusively used for industrial applications with only static or very predictable environmental constraints. These applications require only precomputed trajectories that usually remain unchanged. Recently, however, first articulated robots have gained the mechanical and control capabilities for coping with local uncertainties in their environment and interaction with humans. Powerful and highly sensorized robotic arms and hands as e.g. the DLR Lightweight Robot III (LWR-III) [1], the Barrett WAM Arm [59] and

Figure 1.1: Physical cooperation between humans and robots is desired in future robotic applications and poses the fundamental question of how to ensure safety to the human during such scenarios. As an example a human is interacting with the DLR humanoid *Justin* (left) and the KUKA Lightweight Robot (right). The KUKA/DLR Lightweight Robot is based on the DLR-Lightweight Robot III (LWR-III) and is the result of a technology transfer from DLR to the robot manufacturer KUKA).

the DLR Hand-II [22] were developed over the last decade. Those systems are particularly well designed for applications that incorporate Human-Robot Interaction. Recently, powerful control strategies were developed for this new generation of robots for nominal interaction control and sensitive collision detection and reaction [3, 19, 38, 60, 13]. This significant progress necessitates now the development of powerful real-time collision avoidance methods that are particularly well suited for the robot's demands and capabilities in pHRI tasks. Apart from dealing with a contact alone, the generation of adequate motions that are also able to deal with dynamic environment needs to be treated. In this sense this thesis contributes numerous new insights into reactive motion schemes and outlines several extensions of existing schemes.

In the following we review the major contributions of the present thesis.

Contributions of the Thesis

In this thesis we approach the aforementioned problem by analyzing, selecting, and extending existing strategies and algorithms for generating collision free paths in dynamic environments. For being able to objectively compare

the performance of the chosen schemes (after a pre-selection process) we develop a simulation environment for comparing them in various scenarios with increasing complexity. We extensively evaluate the schemes in numerous 2D, 3D, and 6D scenarios with respect to their ability to generate reactive motions in real-time with limited local knowledge of the possibly dynamic and complex environment. Furthermore, simultaneous goal convergence, while providing coordinated movement in translation and orientation, is a further primary requirement. In general, safe and quick handling of external objects is to be achieved. For implementing such a motion behavior we significantly extend and combine existing collision avoidance algorithms such that they generate smooth, intuitive trajectories and/or virtual disturbance signals that can be fed to low level controllers. We apply the methods to different levels of control and motion generation in order to analyze their respective effectiveness. Furthermore, we provide very promising experimental results for rather complex real-world examples. In particular, we show the feasibility of the algorithms for various scenes as e.g. a static multi-object parcour and the case in which a (dynamically moving) human operator is to be safely circumvented.

In addition to addressing the reactive motion control for pHRI, we also extend the algorithms such that a powerful scheme for physically exploring unknown wire objects with tactile information only is generated. The algorithm is used to explore complex planar 6D wires in both simulation and experiment. It enables the robot to incrementally built a geometric interaction map of the object and updates it according to the respective sensory input.

2 State of the Art

As a basis for reactive motion generation in direct HRI, i.e. in a highly dynamic environment, the basic understanding of robot control, robot perception, and motion generation algorithms is essential. Each layer contributes to fulfill a desired task by offering alternative solutions to a given problem and may serve the other layers accordingly in a supportive manner. The recent advances in robot perception allow a detailed measurement of the current robot state and its environment. Control algorithms that take into account the uncertainties of (particularly) unknown environments as e.g. impedance control allow flexible movement generation.

Further focus lies on the mechanical design of the robot, which directly influences the other layers as well. The robot design determines the kinematic properties as well as the possible motion dynamics. Both need to be taken into consideration in control and motion generation.

In the present chapter we review the state-of-the-art relevant for this thesis. First, a brief overview on collision avoidance in general and some selected application areas is given. Then, an introduction on the soft robotics control schemes of the LWR-III and robot perception in general is given.

2.1 Collision Avoidance

Classical areas of collision avoidance algorithms are unmanned aerial vehicles, mobile robotics, and recently also HRI. In this section an introduction to these applications is given and some existing systems are presented.

Over the last two decades numerous collision avoidance algorithms have been developed. For dynamic environments they are mainly of reactive character, as e.g. Potential Fields [31, 32], Optical Flow [18, 41], Harmonic Potential Fields [43, 30], Circular Fields [53], and Reactive Planning techniques [36]. For static environments probabilistic planning [35, 33] can be used to generate collision free trajectories in complex geometric spaces with multi-DoF robots. In this section we shortly survey several algorithms, whereas the detailed algorithmic descriptions is left for Chapter 3. There, we formally introduce motion generation algorithms, compare them with respect to the applicability to our problem, and contribute extensions for some

Figure 2.1: Parrot Drone [47] (left), Plane UAV [37] (middle), and DLR Quadrocopter [4] (right).

selected algorithms.

2.1.1 Unmanned Aerial Vehicles

Unmanned aerial vehicles (UAVs) are autonomous flying objects as e.g. small aircrafts, helicopters or quadrocopters, see Fig. 2.1. Because these vehicles can reach high speeds and store large amounts of kinetic energy effective local and independent motion generation that directly relies on measured environment date is essential. These systems mostly navigate via GPS and due to payload restriction they have only very limited perception and processing capabilities. Therefore, the calculation capacity for collision avoidance strategies is limited and geometrically complex environment models cannot be used. Thereby, Optical Flow has become one of the most widely used collision avoidance strategies, see Sec. 3.1.5.

2.1.2 Mobile Robots

For complex every day tasks the ability to equip a manipulator with mobility is of great advantage. Typical implementations of such capabilities are on mobile platforms that are equipped with articulated manipulators. Mobility makes it possible to reach objects far beyond the workspace of a single static

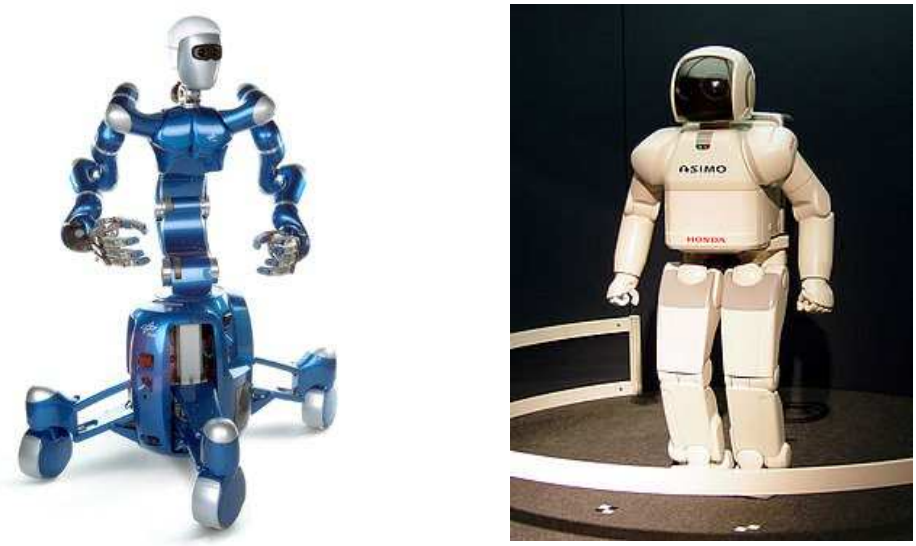


Figure 2.2: DLR Rollin Justin [16] and walking Honda ASIMO [10]

manipulator. Furthermore, transport tasks, such as a "fetch and carry" services can be performed. In addition, a manipulator may benefit from the support of a mobile platform. When e.g. using the platform during the manipulation operation, coordinated whole body control may be executed, which may significantly enhance the task performance. Such schemes can also be adapted for collision avoidance purposes, where e.g. some DoF can be used for platform avoidance, while the manipulator continues to follow the given task [52]. Existing mobile platforms are generally classified into walking and driving systems. Walking systems usually approach the mobility problem by means of legged locomotion. Driving systems on the other hand use wheeled locomotion with various different implementations, see Fig. 2.2 (left). In contrast to the bipedal walking humanoid ASIMO (Fig. 2.2 (right)) a safe stand can e.g. be guaranteed for platforms as the DLR humanoid system Rollin' Justin. Justin consists of a torso, head, and two 7 DoF LWR-III manipulators on a four wheeled platform. In this recent development [16] the wheels are attached to adaptive leg elements in order to gain more flexibility while driving through door hogs and performing manipulation tasks. If a safe stand is e.g. required for the latter task the platform extends its legs and therewith its base, leading to a very steady base.

2.1.3 Human-Robot Interaction

The hardly predictable factor of a human being in the robot workspace is one of the major challenge of future robotics systems. Research institutes all over the world intend to handle this situation as this is still regarded as a major open problem. Especially in industrial scenarios direct interaction between human and robot is, apart from very basic applications, still not being realized. This is to a large extend due to the fact that for these human-robot interaction (HRI) tasks safety issues become the primary concern. Safe robot paths and socially acceptable motions during cooperation with or assistance are aimed for. In such applications unforeseen collisions are of course the main cause of concern and therefore, methods are needed that allow automatic reaction of a robot to physical collisions [19]. Apart from simply reacting to a collision it may also be desirable to perform a task further even during contact, while redundant task DoFs may be used for retracting from the contact [40]. In addition to these reactive control schemes the execution of a task in a human-friendly manner is important. For achieving such behavior [54] introduced the human aware motion planner (HAMP) that takes into account the human standpoint for motion decision in order to ensure

- safe motion that does not harm the human,
- reliable and effective motion tailored to the capabilities of the robot, and
- socially acceptable motion that also takes into account a motion model of the human, as well as his preferences and needs.

First strategies for HAMP with static manipulators are e.g. presented in [34]. Furthermore, adaptive control and motion planning for industrial robots by using visual workspace observation is presented in [17].

A field of HRI that was mainly left out up to now is the reactive motion generation in the presence of humans. However, it is still a major issue to during human presence in the workspace without running into unwanted physical contact. Therefore, in this thesis we extend existing motion algorithms and control methods for approaching a solution for this problem.

In the following we describe control methods that are used throughout this thesis.

Figure 2.3: The DLR Lightweight Robot III.

2.2 Robot Control

In this section we first introduce the rigid body dynamics for stiff robots. Then, we extend this formulation to the flexible joint case, which is especially needed for the robot used in this thesis. They include joint elasticity between motor and link inertia, therefore doubling the state space of the resulting model. An introduction to the used controllers is given and finally, nullspace motion control is described that can be used to enhance Operational space control by independently assigning a certain behavior to the nullspace.

Before going into the details of modeling and control we shortly describe the manipulator we us in this thesis as a reference platform, the DLR Lightweight Robot III (LWR-III).

DLR Lightweight Robot III The LWR-III is especially optimized for its use in space and service robot application, see Fig. 2.3. In contrast to classical industrial robots its electronics, sensors, and drive control are integrated in to the manipulator structure. The main features of the robot are:

- 7 degrees of freedom (DoF) [23],
- Motor & link side position sensors [1],
- Joint torque sensor in each joint,
- Soft robotics control schemes [3],
- Collision detection and reaction in real-time (1 ms) [38].

The LWR-III has an overall weight of 14 kg and a load-to-weight ratio of 1:1. Taking especially the perception and reaction capabilities into account, collisions do not have to be excluded by any means as the robot is able to sensitively react to external forces that act along its structure. On the contrary, the environment information obtained by a contact could be e.g. be used for tactile exploration of the workspace and further movement generation.

2.2.1 Robot Dynamics

The most common methods of deriving the robot dynamics of a stiff manipulator are the Lagrangian formulation or alternatively Newton-Euler algorithm. This leads to the joint space representation of the rigid robot [11].

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}_m. \tag{2.1}$$

$\boldsymbol{\tau}_m \in \mathbb{R}^n$ is the motor torques, $M(\mathbf{q}) \in \mathbb{R}^{n\times n}$ the mass matrix, $C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} \in \mathbb{R}^{n\times n}$ are the centrifugal and Coriolis terms, and $\mathbf{g}(\mathbf{q})\mathbb{R}^n$ is the gravity vector. $\mathbf{q} \in \mathbb{R}^n$ represents the position of the links.

For most industrial robots it is sufficient to use (Equ: 2.1) for expressing the dynamics of the manipulator. For the LWR-III, however, with its intrinsically flexible joints a simple rigid body approach is not accurately enough. The elasticity due to its Harmonic Drive gears and the joint torque sensors needs to be taken into account. The extended new model can be written as [3]

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}_J + D_J K_J^{-1}\dot{\boldsymbol{\tau}}_J - \boldsymbol{\tau}_{\text{ext}} = \boldsymbol{\tau}_{\text{tot}} \tag{2.2}$$

$$B\ddot{\boldsymbol{\theta}} + \boldsymbol{\tau}_J + D_J K_J^{-1}\dot{\boldsymbol{\tau}}_J + \boldsymbol{\tau}_F = \boldsymbol{\tau}_m \tag{2.3}$$

$$\boldsymbol{\tau}_J = K_J(\boldsymbol{\theta} - \mathbf{q}), \tag{2.4}$$

where $K_J \in \mathbb{R}^{n\times n}$ is the diagonal positive definite joint stiffness matrix, $D_J \in \mathbb{R}^{n\times n}$ the diagonal positive definite joint damping matrix, $\boldsymbol{\theta} \in \mathbb{R}^n$

the motor side position, and $B \in \mathbb{R}^{n \times n}$ the diagonal positive definite motor inertia matrix.

Next, we introduce the LWR-III control principle for Cartesian impedance control. For further description of flexible joint robots, please refer to [39].

2.2.2 Cartesian Impedance Control

In general, implemented controller types for the LWR-III are impedance control, admittance control, stiffness control, position control, and torque control with gravity compensation. Since we use exclusively Cartesian impedance control in this thesis, we only review the theory of this particular control type. In general, Cartesian impedance control aims at the implication of a certain disturbance response of the robot with respect to external contact wrenches.

$$\mathbf{F}_{\text{ext}} = M_x \ddot{\tilde{\mathbf{x}}} + D_x \dot{\tilde{\mathbf{x}}} + K_x \tilde{\mathbf{x}}, \tag{2.5}$$

where M_x, D_x and K_x are the desired Cartesian inertia, damping, and stiffness of the system and $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_d$ with $\tilde{\mathbf{x}} \in \mathbb{R}^m$. Therefore, the impedance controlled robot behaves as a generalized mass-spring-damper system. Such dynamics can be imprinted in joint or Cartesian space. For describing the particular implementation of Cartesian impedance control in the LWR-III some preliminaries concerning the underlying joint space controllers are reviewed next .

As the LWR-III is able to directly measure the joint torques $\boldsymbol{\tau}_J$, a low-level torque control loop is implemented [2], which especially performs high-performance joint level vibration damping and the torque interface for higher level control schemes. The overall controller can be written as

$$\boldsymbol{\tau}_m = B B_\theta^{-1} \mathbf{u} + (I - B B_\theta^{-1}) \boldsymbol{\tau}_J + D_J K_J^{-1} \dot{\boldsymbol{\tau}}_J \tag{2.6}$$

$$\mathbf{u} = -K_\theta (\boldsymbol{\theta} - \boldsymbol{\theta}_d) - D_\theta \dot{\boldsymbol{\theta}} + \bar{\mathbf{g}}(\boldsymbol{\theta}), \tag{2.7}$$

where $B_\theta \in \mathbb{R}^{n \times n}$, $K_\theta \in \mathbb{R}^{n \times n}$ and $D_\theta \in \mathbb{R}^{n \times n}$ are the diagonal positive definite desired, motor inertia, stiffness, and damping matrix. $\boldsymbol{\theta}_d \in \mathbb{R}^n$ is the desired joint configuration and $\bar{\mathbf{g}}(\theta) \in \mathbb{R}^n$ the gravity compensation term. The new (torque) input variable $\mathbf{u}$ is given by the desired joint position controller. With $\tilde{\boldsymbol{\theta}} =: \boldsymbol{\theta} - \boldsymbol{\theta}_d$ the closed loop control system can be written as

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}_{\text{ext}} = \boldsymbol{\tau}_J + D_J K_J^{-1} \dot{\boldsymbol{\tau}}_J \tag{2.8}$$

$$B_\theta \ddot{\boldsymbol{\theta}} + \boldsymbol{\tau}_J + D_\theta \dot{\boldsymbol{\theta}} + K_\theta \tilde{\boldsymbol{\theta}} = \bar{\mathbf{g}}(\boldsymbol{\theta}), \tag{2.9}$$

with the last term being equivalent to

$$\mathbf{u} = B_\theta \ddot{\boldsymbol{\theta}} + \boldsymbol{\tau}_J. \tag{2.10}$$

Furthermore, (2.9) can be rewritten in the form of a joint feedback controller

$$\boldsymbol{\tau}_m = -K_P \tilde{\boldsymbol{\theta}} - K_D \dot{\boldsymbol{\theta}} + K_T(\bar{\mathbf{g}}(\boldsymbol{\theta}) - \boldsymbol{\tau}_J) - K_S \dot{\boldsymbol{\tau}}_J + \bar{\mathbf{g}}(\boldsymbol{\theta}), \tag{2.11}$$

with

$$\begin{array}{rcl} K_P & = & B B_\theta^{-1} K_\theta \\ K_D & = & B B_\theta^{-1} D_\theta \\ K_T & = & B B_\theta^{-1} - I \\ K_S & = & B B_\theta^{-1} K_{s0} K_J^{-1} - D_J K_J^{-1}. \end{array} \tag{2.12}$$

In this thesis desired motions are expressed in Cartesian coordinates $x(\bar{\mathbf{q}})$, requiring Cartesian impedance control. The described joint impedance control law is now used as an interface for the outer Cartesian impedance loop. The control loop for the desired Cartesian behavior is

$$\mathbf{u} = -J(\bar{\mathbf{q}})^T (K_x \tilde{\mathbf{x}}(\bar{\mathbf{q}}) + D_x \dot{\mathbf{x}}(\bar{\mathbf{q}})) + \bar{\mathbf{g}}(\boldsymbol{\theta}) \tag{2.13}$$

$$\tilde{\mathbf{x}}(\bar{\mathbf{q}}) = \mathbf{x}(\bar{\mathbf{q}}) + \mathbf{x}_d \tag{2.14}$$

$$\dot{\mathbf{x}}(\bar{\mathbf{q}}) = J(\bar{\mathbf{q}}) \dot{\boldsymbol{\theta}}, \tag{2.15}$$

where $K_x \in \mathbb{R}^{m \times m}$ is the desired Cartesian stiffness matrix, $D_x \in \mathbb{R}^{m \times m}$ the desired damping matrix $\mathbf{x}_d \in \mathbb{R}^m$ the desired Cartesian tip pose, and $\mathbf{x}(\bar{\mathbf{q}}) = f(\bar{\mathbf{q}})$ the position and orientation calculated by the forward kinematics map f. From the derivation of the kinematics map the Jacobian $J(\bar{\mathbf{q}}) = \frac{\partial f(\bar{\mathbf{q}})}{\partial \bar{\mathbf{q}}}$ is directly computed and $\bar{\mathbf{q}} = h^{-1}(\theta)$ is the static equivalent of $\mathbf{q}$. Finally, using (2.10) and (2.13) the Cartesian impedance control law can be written as

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}_{\text{ext}} = \boldsymbol{\tau}_J + D_J K_J^{-1} \dot{\boldsymbol{\tau}}_J \tag{2.16}$$

$$B_\theta \ddot{\boldsymbol{\theta}} + J(\bar{\mathbf{q}})^T (K_x \tilde{\mathbf{x}}(\bar{\mathbf{q}}) + D_x \dot{\mathbf{x}}(\bar{\mathbf{q}})) + \boldsymbol{\tau}_J + D_J K_J^{-1} \dot{\boldsymbol{\tau}}_J = \bar{\mathbf{g}}(\boldsymbol{\theta}) \tag{2.17}$$

$$\tau_J = K_J(\boldsymbol{\theta} - \mathbf{q}). \tag{2.18}$$

After this introduction to Cartesian impedance control for the LWR-III we describe some treatment of nullspace motion in the following.

2.2.3 Nullspace Motion

For controlling the full operational space motion of the LWR-III we need $m = 6$ DoF. Since the robot has $n = 7$ DoF, the remaining DoF $r = n - m = 1$ can be used for nullspace motion. A common approach to use the redundant degree(s) is to define $\mathbf{x}$ together with auxiliary variables, the nullspace coordinates. To construct then the so called nullspace base matrix $Z(\mathbf{q}) \in \mathbb{R}^{r \times n}$ needs to be computed. For the general case the Jacobian matrix can always be partitioned by reordering it $J(\mathbf{q}) = [J_m(\mathbf{q})\ J_r(\mathbf{q})]$, where $J_m(\mathbf{q}) \in \mathbb{R}^{m \times m}$ is at least invertible. Analogously, the nullspace matrix can always be partitioned as $Z(\mathbf{q}) = [Z_m(\mathbf{q})\ Z_r(\mathbf{q})]$, with $Z_m(\mathbf{q}) \in \mathbb{R}^{r \times m}$ and $Z_r(\mathbf{q}) \in \mathbb{R}^{r \times r}$. Therefore, $J(\mathbf{q})Z(\mathbf{q})^T = \mathbf{0}$ has to be fulfilled and $Z(\mathbf{q})$ has full row rank. With this partitioning, the condition $J(\mathbf{q})Z(\mathbf{q})^T = \mathbf{0}$ can be written as

$$J_m(\mathbf{q})Z_m(\mathbf{q})^T + J_r(\mathbf{q})Z_r(\mathbf{q})^T = \mathbf{0}. \tag{2.19}$$

One possible solution of this equation is given by the particular choice

$$Z_m(\mathbf{q})^T = -J_m(\mathbf{q})^{-1}J_r(\mathbf{q}), \tag{2.20}$$

$$Z_r(\mathbf{q})^T = I, \tag{2.21}$$

which by construction gives a full rank nullspace base matrix.

As described in [46], the approach of defining $J(\mathbf{q}) = \left[-J_r(\mathbf{q})^T J_m(\mathbf{q})^{-T}\ I\right]$ can be scaled by $\det(J_m(\mathbf{q}))$ to

$$\bar{Z} = \left[-J_r(\mathbf{q})^T \mathrm{adj}(J_m(\mathbf{q}))^T\ \det(J_m(\mathbf{q}))I\right], \tag{2.22}$$

whereby the Jacobian inverse does not have to be computed. Furthermore, this general approach is characterized by the fact that the quadratic kinematic manipulability measure is $m^2_{\mathrm{kin}}(\mathbf{q}) := \det(J(\mathbf{q})J(\mathbf{q})^T) = \det(\bar{Z}(\mathbf{q})\bar{Z}(\mathbf{q})^T)$.

For a one-dimensional nullspace the nullspace matrix simplifies to the bijective vector $\mathbf{z}(\mathbf{q})$ with

$$z_i(\mathbf{q}) = (-1)^{n+i}\det(J_i(\mathbf{q})). \tag{2.23}$$

After this description of the nullspace, we continue with the implementation of null-space motion. By use of virtual or real external forces, as e.g. described in [40], this approach can be used for realizing collision avoidance in the null-space, while keeping the desired task space motion. Hereby, Cartesian forces are orthogonally projected into the nullspace with respect

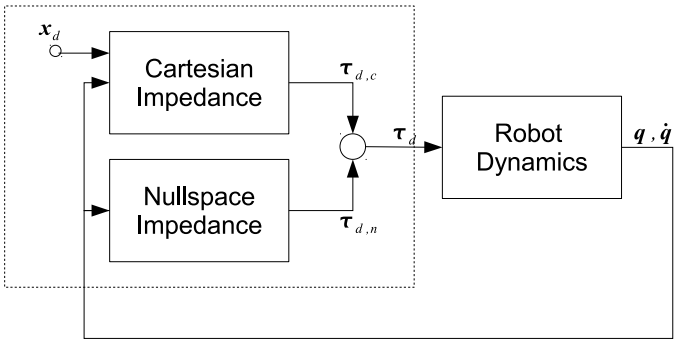


Figure 2.4: The nullspace impedance control principle.

to the actual dynamics. In [46] the basic approach for such a control schemes based on a torque interface is described. Generally speaking, the introduced scheme leads to the block diagram depicted in Fig. 2.4.

The approach is such that the desired robot torque $\boldsymbol{\tau}_d$ consists of a Cartesian impedance torque $\boldsymbol{\tau}_{d,c}$ for the end-effector and a Nullspace impedance torque $\boldsymbol{\tau}_{d,n}$.

$$\boldsymbol{\tau}_d = \boldsymbol{\tau}_{d,c} + \boldsymbol{\tau}_{d,n} \tag{2.24}$$

To define $\boldsymbol{\tau}_{d,n}$, position, velocity and external or virtual forces are transformed into joint variables. This is done via the inverse kinematics map $f(\mathbf{x})^*$ or the Jacobian $J(\mathbf{q}) = \frac{\partial f(\mathbf{q})}{\partial \mathbf{q}}$.

$$\mathbf{q}_{d,0} = f(\mathbf{x}_{d,0})^* \tag{2.25}$$

$$\dot{\mathbf{q}}_{d,0} = J(\mathbf{q})^T \dot{\mathbf{x}}_{d,0} \tag{2.26}$$

$$\boldsymbol{\tau}_{d,0} = J(\mathbf{q})^T \mathbf{F}_{\text{ext/vir}} \tag{2.27}$$

To what extent $\mathbf{q}_{d,0}$ and $\dot{\mathbf{q}}_{d,0}$ can be reached depends on $\mathbf{x}_d$ and its possible joint representations. In the following, $\boldsymbol{\tau}_{d,0}$ represents the into nullspace projected torque. For possibilities to handle the ratio of Cartesian and nullspace impedance and target function please refer to [46]. Furthermore, when considering $\mathbf{q}_{d,0}$ and $\dot{\mathbf{q}}_{d,0}$, we may define an underlying joint space impedance behavior as follows.

$$\boldsymbol{\tau}_{d,0} = -D_n(\dot{\mathbf{q}} - \dot{\mathbf{q}}_{d,0}) - K_n(\mathbf{q} - \mathbf{q}_{d,0}), \tag{2.28}$$

where $K_n \in \mathbb{R}^{n\times n}$ and $D_n \in \mathbb{R}^{n\times n}$ are symmetric and positive definite matrices that represent a desired stiffness and damping. Because $\boldsymbol{\tau}_{d,0}$ cannot be used directly it has to be projected into the Nullspace for example by the projection matrix $P(\mathbf{q})$.

$$\boldsymbol{\tau}_{d,n} = P(\mathbf{q})\boldsymbol{\tau}_{d,0} \tag{2.29}$$

Hereby, the projection matrix $P(\mathbf{q})$ is a counter piece of $J(\mathbf{q})^T$ and can be defined in multiple ways. Three examples are shown in [46]. The aim is to generate a $\boldsymbol{\tau}_n$ that is orthogonal to the Cartesian torque $\boldsymbol{\tau}_c = J(\mathbf{q})^T\mathbf{F}_{\text{imp}}$. Therefore, the projection matrix also has to be orthogonal to the Cartesian impedance configuration. Then, its changes do not affect the fulfillment of the task. The projection matrix can e.g. be defined by using the nullspace vector (one dimensional nullspace) and can be written as

$$P(\mathbf{q}) = M(\mathbf{q})Z(\mathbf{q})^T(Z(\mathbf{q})M(\mathbf{q})Z(\mathbf{q})^T)^{-1}Z(\mathbf{q}). \tag{2.30}$$

Furthermore, the described methodology can be used to generate also more complex nullspace behavior as e.g. to scale the influence of external or virtual forces by minimal, partial, or full task relaxation, as e.g. shown in [40].

2.3 Robot Perception

Robotics has evolved to a system integration engineering field, as e.g. defined by M. Brady [7]: The intelligent connection of perception to action. This means that for performing adequate action, as e.g. the generation of sensor based reaction motions, it is elementary for a robot to perceive its environment well enough to make autonomous decisions. Robot perception collects environmental information and combines various multi-modal sensorial information. This may be used to identify or assign particular sensorial stimuli to certain objects of the environment.

In this section it is described what information robotic systems are possibly able to perceive regarding their internal states and external environment. The used structure and classification is influenced by [49, 12, 48]. The items, which are signed by *-mark in Tab. 2.1, Tab. 2.2 or Fig. 2.5 indicate the sensor types that are used in the LWR-III and the Co-Worker setup (see Sec.4.1 and 4.1), which is the reference platform of this thesis.

In a very general sense, robot perception is performed by various sensor systems. They convert physical (analog) quantities, or if possible their derivates

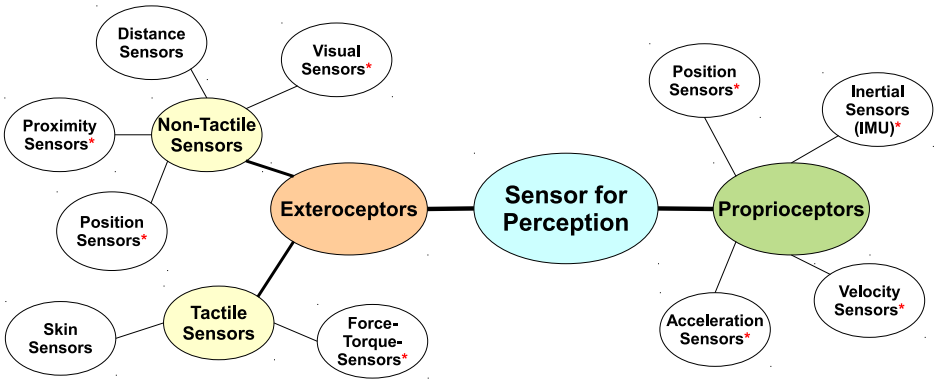


Figure 2.5: Classification of robot perception sensors.

(e.g. velocity) to processable electronic (analog) signals. In general, sensor systems can be classified according to

- physical units (e.g. position, force, velocity),
- physical principles (e.g. Hall-principle),
- technologies (e.g. electric, mechanical, magnetic), and
- typification
  - internal (measure robot internal variables) and external (measure the state of the robot environment),
  - with contact, without contact
  - active (e.g. ultrasound, infrared, laser rangefinders) or passive (e.g. tactile sensors, photo detectors, microphones, cameras without headlight) sensors.
  - and character as tactile sensors, proximity sensors, distance sensors, visual sensors and position sensors

In this thesis the type classification as depicted in Fig. 2.5 is chosen as the reference. We distinguish “exteroceptors”, or external, and “proprioceptors” or internal sensors. Internal sensors are used to identify the internal robot state. External sensors can be separated into tactile sensors and non-tactile sensors. Tactile sensors collect contact information by force-torque-pressure data and non-tactile sensors gather proximity, distance, position, or general visual data for creating an environment representation.

2.3.1 Proprioception

Internal Sensors			
Position Sensors	**Velocity Sensors**	**Acceleration Sensors**	**Force-Torque Sensors**
potentiometer resolver optical encoder* magnetic*	tachometer magnetic pickup	SI-sensors transducers optical capacitive	strain gages on profiles* strain gages on links

Table 2.1: Overview of "proprioceptors", or internal sensors.

The control of a robot potentially utilizes knowledge about various kinematic and dynamic variables to move a series of links and actuate joints. Especially the relative movement of links and the forces acting on the structure of the robot need to be observed for generating motor torques so that the articulated mechanical structure performs a desired motion. Proprioceptors or internal sensors are sensors measuring both, kinematic and dynamic variables of the robot.

The commonly measured kinematic variables are joint positions. Velocities are usually obtained by numerical differentiation, however, can be measured as well (please note that acceleration is usually not obtainable anymore by differentiating position due to sensor noise and quantization effects). For the proper control of the manipulator dynamics, quantities as forces and torques are equally important to be monitored. Options to measure these units are listed in Tab. 2.1.

Position Sensors

The most common joint angle position sensors are potentiometers, synchros, inductosyns, resolvers, and encoders. For computer interfaces encoders as e.g. digital position transducers are the most common devices.

Incremental encoders measure the relative-position and generate pulses that are proportional to angular velocity. They are less expensive and offer a higher resolution than absolute encoders. Incremental encoders, however, have to be initialized by moving them in a reference ("zero") position after power loss. Absolute encoders, which also do not accumulate sensor errors as incremental encoders may do, simply have to read a distinct n-bit code that describes the current position.

Joint position sensors are usually mounted on the motor shaft. When being directly attached to the joint, position sensors allow position feedback to the controller that is affected by the joint backlash and drive train compliance. Additionally, an internal drive position sensor can be performed by Hall effect sensors, which detect the drive rotor position by creating potentials that result from a current trough an electromagnetic field and the occurring Lorentz forces.

Velocity Sensors

Angular velocity is usually measured (when not calculated by direct differentiation of joint position) by tachometer transducers. The tachometer generates a DC voltage that is proportional to the shaft rotational speed. Current digital tachometers in robotic applications use magnetic pickup sensors and no DC motor-like tachometers anymore, because the former are significantly smaller.

Acceleration Sensors

Acceleration sensors are based on Newton's second law and measure the fictitious force that act on the known accelerated mass. Different types of acceleration transducers exist: stress-strain gage, piezoelectric, capacitive, and inductive. Recent micro-mechanical accelerometers measure the force by the strain in elastic cantilever beams that are formed from silicon dioxide in an integrated circuit fabrication process.

Force-Torque Sensors

The most common method to measure forces and torques are strain gages mounted on specially profiled shafts (square, crosswise or radial beam). Sometimes, strain gages are also mounted on the manipulator links where they are used to estimate the flexibility of the mechanical structure and external torques.

2.3.2 Exteroception

Exteroceptors are sensors that measure the positional or force-type interaction of the robot with respect to its environment. Exteroception is mainly of non-tactile character, however, tactile information may also serve as external perception. An overview of external sensors is given in Tab. 2.2.

External Sensors				
	Non-Tactile			
Tactile Sensors	**Distance Sensors**	**Visual Sensors**	**Proximity Sensors**	**Position Sensors**
skin force-torque*	optical*	stereo camera CCD photodiodes camera*	acoustic optical* capacitive induction	infrared* GPS

Table 2.2: Overview "exteroceptors", or external sensors.

Tactile Sensors

Tactile or contact sensors are used to detect contact between two mating parts or to measure the interaction forces and torques that appear between the robot and the environment. This could be e.g. the grasping forces during manipulation tasks. Another type of contact sensors are skin sensors, which measure a multitude of parameters along the touched object surface. Both types can be applied for tactile exploration [26, 6].

Force-Torque Sensors Interaction forces and torques during mechanical assembly operations at the robot hand level can be measured by sensors that are mounted in the joints (then this is measured by interceptive sensing) or on the manipulator wrist. A solution on joint level is not that attractive for traditional industrial robots because it needs a conversion of the measured joint torques to equivalent forces and torques at the end-effector level (this is due to model uncertainties and typical sensor equipment of industrial robots not really possible). The external forces and torques $\mathbf{F}_{\text{ext}}$ measured by a wrist sensor can be converted directly by $\boldsymbol{\tau}_{\text{ext}} = J^T(\mathbf{q})\mathbf{F}_{\text{ext}}$ to joint level. Wrist sensors, however, are sensitive, small, compact and relatively light.

Robots, on the other hand, which already use internal force-torque sensors for control do not need any additional external devices as e.g. shown in [20].

Tactile Skins Tactile skins continuously sense variable contact forces over an area with a certain spatial resolution. Skin sensing is more complex than touch sensing, which is usually a simple vectorial force-torque measurement at a single point. Skin sensors mounted on the end-effector allow the robot to measure contact force profiles and slippage. A good overview on this sensing principles is e.g. given in [25].

Tactile sensor technologies are conductive elastomers, strain gage based, piezoelectric, capacitive, and optoelectronic ones. These technologies can be further grouped by their operating principles into two categories: force-sensitive and displacement-sensitive. The force-sensitive sensors (conductive elastomer, strain gage, and piezoelectric) measure contact forces, while the displacement-sensitive (optoelectronic and capacitive) sensors measure the mechanical deformation of an elastic overlay, therefore provide significant mechanical filtering as well.

Generally, tactile exploration is the result of a complex exploratory perception act by the combination of proprioception and exteroception with two distinct modes. First, passive sensing produced by skin sensory network provides information about contact force and the contact geometric profile. Second, active sensing integrates the skin sensory information with proprioceptive sensory information as joint positions and velocities.

Non-Tactile Sensors

There are three types of non-tactile sensors:

1. proximity sensors,
2. range sensors,
3. and vision sensors.

The first two can be distinguished by their effective range. Vision on the other hand is able to measure multiple features as e.g. color or texture.

Proximity Sensors Proximity sensors detect the presence of nearby objects without touching them. Therefore, these sensors can be used for near-field robotic operations. Proximity sensors are classified according to their operating principle into inductive, hall effect, capacitive, sonar, and optical.

Inductive sensors are based on the change of inductance due to the presence of metallic objects. Hall effect sensors are based on the relation between the voltage in a semiconductor material and the magnetic field across this material. Inductive and Hall effect sensors detect only the proximity of ferromagnetic objects. Capacitive sensors are potentially capable of measuring the proximity of any type of solid or liquid materials. Sonar and optical sensors are based on the sensed modification of emitted signals by objects in their proximity.

Range and Position Sensors Range sensors measure the distance to objects in their operational range. They are used for robot navigation, obstacle avoidance, or to recover the third dimension for monocular vision. Range sensors are based on one of the two principles: time-of-flight or triangulation.

Time-of-flight sensors estimate the range by measuring the time elapsed between the transmission and return of a pulse. Laser-range finders and sonar are the best known sensors of this type.

Triangulation sensors measure the range by detecting a given point on the object surface from two different points of view at a known distance from each other. Knowing this distance and the two view angles from the respective points to the aimed surface point, a simple geometrical operation yields the range. For a more complete overview of applications as well as measuring systems with artificial arms, please refer to [51].

Infrared tracking systems as e.g. the ARTtrack2 use infrared optical tracking cameras with active or passive markers. They measure postures in the robot workspace with high accuracy.

Typical position sensors for outdoor applications are GPS position sensors, possibly reaching high accuracy localization for large scale areas. These sensors are well suited for mobile robotics and cannot be used for indoor applications. However, there exist applications, where robots that are working inside and outside use multiple combinations of sensors as GPS, ranging sensors, and visual sensors depending on their current operation range.

Visual Sensors Robot visual perception is a complex sensing process and offers numerous opportunities for environmental perception. The procedures involve extracting, characterizing, and interpreting of information from images in order to describe or only identify objects.

A vision sensor as a camera converts the visual information into electrical signals, which are then sampled and quantized by a special computer interface electronics, yielding a digital image. Most existing visual sensors are designed for television or personal computer, which is not necessarily optimally suited for robotic applications. Currently, there is a multitude of commercial computer interfaces, providing images with standard TV videorate of 30 Hz.

Every image processing software consists of following steps: pre-processing, segmentation, description, recognition and interpretation [55]. Pre-processing techniques usually deal with noise reduction and detail enhancement. Algorithms as edge detection or region growing are used for segmentation, i.e.

to extract objects from a particular scene. The objects are described by preferably invariant features. For recognition tasks these features are used to classify the object. For interpretation a particular meaning is assigned to the ensemble of recognized objects.

There are several other possibilities to update the visual perception, e.g. by CCD (Charge Coupled (Area Imaging) Device) image sensors, structured lighting with special light stripes, grids or other patterns and stereo cameras.

3 Algorithmic Foundations

In this chapter we give an overview on existing motion generation algorithms and discuss their particular capabilities for online collision avoidance. We define objective criteria for comparing the schemes and perform a selection based on their suitability. Then, we discuss significant extensions we developed for the chosen algorithm, which improve the corresponding behavior and runtime.

3.1 Motion Generation Algorithms

Motion generation algorithms can be classified into three categories (see Fig.3.1):

1. reactive,
2. planning based,
3. and reactive planner based.

In this section we describe following reactive motion generation methods: Potential Fields, Human Navigation Potential Fields, Harmonic Potential Fields, Circular Fields, and Optical Flow. Furthermore, we review reactive and probabilistic planners and discuss their applicability for the chosen problem. Elastic Strips are considered as a combination of planning and reactive methods. In the following description, we use the term "avoiding objects" for elements that shall avoid obstacles, e.g. the robot or parts of it. In turn "obstacle objects" are objects that have to be avoided.

Reactive schemes often use physical analogies for generating trajectories in real-time. Many of them, as e.g. Potential Fields are force based and use a point mass model that is driven by certain forces as the associated reference command to the robot. The point mass is considered to be inertia-free and to be independent from gravity. Therefore, its motion can be described as

$$M_x \ddot{\mathbf{x}} = \mathbf{F}_d = \mathbf{F}_a + \mathbf{F}_D + \sum_j \mathbf{F}_{\mathrm{ob},j} \tag{3.1}$$

$$\ddot{\mathbf{x}} = M_x^{-1} \mathbf{F}_d, \tag{3.2}$$

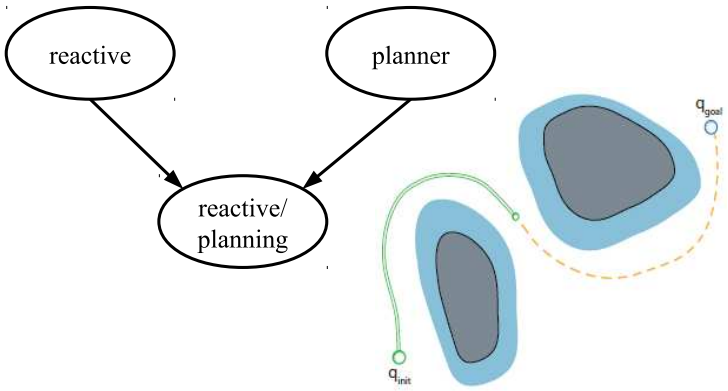


Figure 3.1: Classification of collision avoidance algorithms.

with $M_x = m\mathbf{I} \in \mathbb{R}^{3\times3}$ being the point mass, $\mathbf{F}_d \in \mathbb{R}^3$ the desired forces acting on the point mass, and $\ddot{\mathbf{x}} \in \mathbb{R}^3$ the resulting acceleration. The desired force input usually consists of an attractive force $\mathbf{F}_a$, a damping force $\mathbf{F}_D$, and a virtual force component,which is the sum of obstacle forces $\mathbf{F}_{\text{ob},j}$.

Alternativly, one can steer motion by a so called gradient dynamical system in Cartesian space [43]. The unified system dynamics may be written as

$$\ddot{\mathbf{x}} = -b\dot{\mathbf{x}} - \nabla\Phi(\mathbf{x}) \tag{3.3}$$

and analoguesly for angular space as [27]

$$\ddot{\boldsymbol{\phi}} = -b\dot{\boldsymbol{\phi}} - \nabla\Phi(\boldsymbol{\phi}). \tag{3.4}$$

In (3.3 and 3.4) b is a damping constant and $\boldsymbol{\phi} \in \mathbb{R}^3$ is the motion direction described by a freely chosen angular base.

After this general principle description we look more closely at several reactive methods.

3.1.1 Potential Fields

The Potential Field (PF) approach is motivated by the physical analogon of a charged particle that moves through an electrostatic field. The desired destination is represented by an attractive field. The structure of the algorithm is quite simple and can be implemented in a feedback form, so that it may run in the inner most control loop. A possible choice for the static obstacle

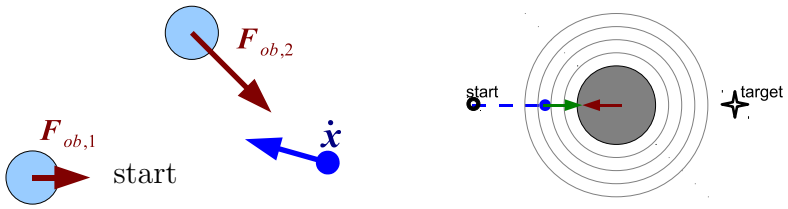


Figure 3.2: Principle of PFs (left) and it simplest local minimum (right).

potential [56] is e.g.

$$U_{x_j} = -\frac{1}{2} k_r \frac{1}{\|\mathbf{x} - \mathbf{x}_{n_j}\|^2}. \tag{3.5}$$

For spherical objects this leads to repulsive forces of the form

$$\mathbf{F}_{ob,j} = -\nabla U_{x_j} = k_r \frac{\mathbf{x} - \mathbf{x}_{n_j}}{\|\mathbf{x} - \mathbf{x}_{n_j}\|^3}, \tag{3.6}$$

where k_r is the repulsive factor. $\mathbf{x}$ is the position of the point mass, and $\mathbf{x}_{n_j}$ the position of obstacles j, parts of the obstacle, or the obstacle clusters. The inverse square dependency ensures that a collision is avoided and a closer obstacle results in an increasing repulsive force, see Fig. 3.2 (left).

A similar approach, which we use in this thesis, is to define a repulsive force that increases at slower rates towards the obstacle:

$$\mathbf{F}^*_{ob,j} = k_r \frac{\mathbf{x} - \mathbf{x}_{n_j}}{\|\mathbf{x} - \mathbf{x}_{n_j}\|^2} \tag{3.7}$$

This results in a similar behavior to (3.1.1), however, it needs fewer operations to be calculated.

A general problem with the PF approach is that the robot may get stuck in local minima. They usually result from the superposition of subfields of multiple obstacles or due to a field that is associated with a geometrically complex (non-convex) object. The most trivial deadlock can even be achieved by an anti-parallel spherical obstacle and goal attraction forces, see Fig. 3.2 (right). However, due to discretization this situation is most likely to be avoided in reality. Nonetheless, already simple symmetrical constellation of two obstacles will generate a new local minimum. In [31, 32] a scheme is introduced for avoiding local minima for spherical objects with potential

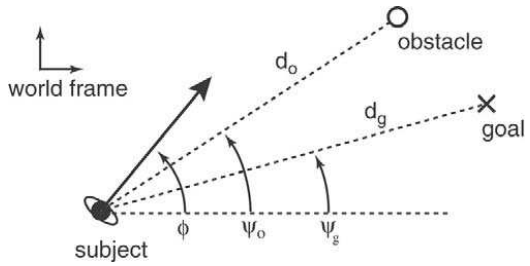


Figure 3.3: Angle and distance information for HNPF [27].

navigation functions. However, the extension for random obstacles is still an open problem.

In the following we describe an adaptation of the potential fields approach, the human navigation potential fields.

3.1.2 Human Navigation Potential Fields

Human inspired navigation principles in connection with Potential Fields were introduced in [27]. They generate a motion based on the angle to obstacles ψ_O and goal ψ_g, see Fig. 3.3. Furthermore, the subject orientation ϕ, the distance to the goal d_g, and the distance to obstacle d_O are influencing the desired trajectory, which is basically an angular motion path.

$$\ddot{\phi} = \frac{dV}{d\phi} - b\dot{\phi} \qquad \text{with} \quad V(\phi) = V_g[\psi_g, d_g](\phi) + \sum_j V_{O,j}[\psi_{O,j}, d_{O,j}](\phi) \tag{3.8}$$

The velocity decreases exponentially with respect to the proximity to the obstacles in order to avoid collisions.

$$\dot{\mathbf{x}} = \max\{\dot{\mathbf{x}}_{\max} e^{k_v V_O} - \epsilon, 0\}, \tag{3.9}$$

where ϵ is a small positive constant, k_v is the gain for the obstacle potential V_O so that the velocity becomes zero for sufficiently large values of V_O. With this potential design the system has of course a local minimum as the speed approaches zero when encountering large obstacle potentials. This directly leads to the observation that e.g. closed U-objectcluster cannot be avoided, see Fig. 3.4.

Next, we discuss Harmonic Potential Fields, which is an interesting class of motion generation algorithms that do not suffer the local minimum problem.

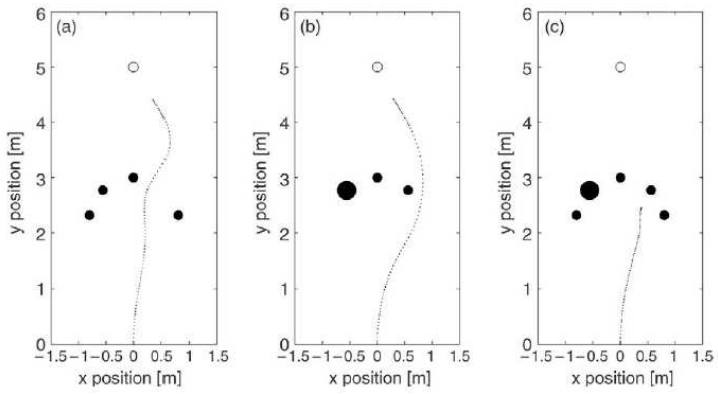


Figure 3.4: An U-Objectcluster may create local minimum for the HNPF.

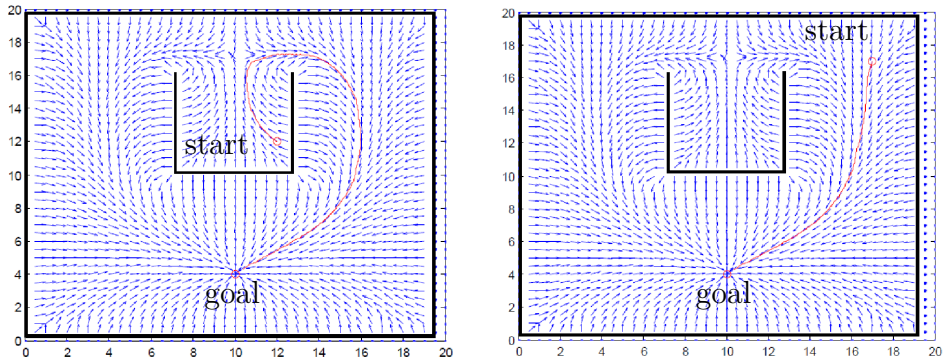


Figure 3.5: Example of HPF gradient field [14].

3.1.3 Harmonic Potential Fields

The Harmonic Potential Fields approach (HPFs) was proposed initially in [45]. Its main idea is to create a potential field without local minima. This is done by using the Laplace equation. The physical analogon is an incompressible fluid that generates a velocity field with vanishing vorticity.

$$\nabla^2 \Phi(\mathbf{x}) \equiv 0 \qquad \mathbf{x} \in \mathbb{R}^3. \tag{3.10}$$

From (3.10) we obtain the gradient dynamical system

$$\dot{\mathbf{x}} = -\nabla \Phi(\mathbf{x}) \qquad \mathbf{x}(0) = \mathbf{x}_0 \in \mathbb{R}^3. \tag{3.11}$$

The result is a velocity field without local minima that guides the robot to the goal configuration. For better understanding a simple example is given in

Fig. 3.5 for the dynamical system of (3.10). It depicts a 2D U-trap scenario and the generated gradient field with a red trajectory. This leads from the upper start position to the goal at the bottom.

The negative gradient may e.g. be used as a driving force by introducing dynamic system associated with a 1 kg point mass and a damping component. This yields

$$\ddot{\mathbf{x}} = -b_D \dot{\mathbf{x}} - \frac{\partial \Phi}{\partial \mathbf{x}}. \tag{3.12}$$

In [43] multiple methods are described to integrate HPFs into dynamical systems, using complex damping components and HPFs to generate an virtual Cartesian driving force

$$\mathbf{F} = -b_D h(\mathbf{x}, \dot{\mathbf{x}}) - k_r \nabla \Phi(\mathbf{x}). \tag{3.13}$$

There are already some published examples for 3D motion with HPFs as described in [30] for the 3D motion of an line element in a 2D environment.

One of the main drawbacks of HPFs is that despite their lack of local minima they cannot be directly used for generating virtual driving forces. This is due to the fact that the introduction of virtual dynamics can lead to collisions with the environment.

Next, we describe another local minima free algorithm, the Circular Fields approach.

3.1.4 Circular Fields

The Circular Fields approach (CFs) is based on the generation of artificial electro-magnetic-fields (B-Fields) that are generated by virtual current elements associated to the surface of obstacles. Thus, in contrast to electrostatic charges for PFs the analogon of dynamical electric charges are consulted. This adaptation of a B-Field $\mathbf{B}$ generates obstacle forces $\mathbf{F}_{ob,j}$ that are vertical to the avoiding object velocity vector $\dot{\mathbf{x}}$ [53], see Fig. 3.6.

$$\mathbf{F}_{ob,j} = \dot{\mathbf{x}} \times \sum_i \mathbf{B}_i \tag{3.14}$$

The generated forces cause re-orientation of the instantaneous velocity vector as well as yield a guidance of the avoiding object around the obstacle object j. This process is done without dissipating energy from the system.

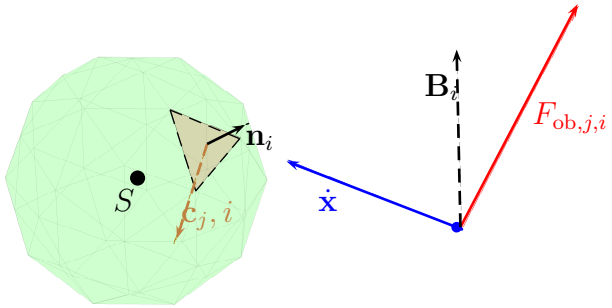


Figure 3.6: Principle of the Circular Fields for the example of a spherical obstacle.

The local circular field $\mathbf{B}_i$ of each surface element i, acting on the virtual particle is defined as

$$\mathbf{B}_i = I_K \frac{\mathbf{c}_{j,i} \times \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|}}{l_i^2} \mathrm{d}a_i, \tag{3.15}$$

where I_K is the virtual current factor, $\mathbf{c}_{j,i}$ is the current direction vector of surface element i, $l_i = \|\mathbf{x} - \mathbf{x}_{\mathbf{n}_i}\|$ is the distance of the current position of the point mass $\mathbf{x}$, and $\mathbf{x}_{\mathbf{n}_i}$ is the position of the obstacle surface element.

Proof of Goal Convergence

For the Circular Field approach a quite straight forward proof for goal convergence can be derived as follows. The point mass dynamic system equation is

$$m\ddot{\mathbf{x}} = -k_a(\mathbf{x} - \mathbf{x}_d) + \dot{\mathbf{x}} \times \sum_i \mathbf{B}_i - k_d\dot{\mathbf{x}}, \tag{3.16}$$

where the quadratic potential field for goal attraction is

$$U = \frac{1}{2} k_a (\mathbf{x} - \mathbf{x}_d)^T (\mathbf{x} - \mathbf{x}_d). \tag{3.17}$$

An energy-like Lyapunov Function can be found as

$$\mathcal{V} = \frac{m}{2} \dot{\mathbf{x}}^T \dot{\mathbf{x}} + U(\mathbf{x}). \tag{3.18}$$

Here $-k_a(\mathbf{x} - \mathbf{x}_d) = \mathbf{F}_a$ is the goal attractor, which is the partial time derivative of U. $k_d\dot{\mathbf{x}}$ is the damping term and $\dot{\mathbf{x}} \times \sum_j \mathbf{B}_j$ is the sum of obstacle forces. Calculating the derivative of $\mathcal{V}$

$$\dot{\mathcal{V}} = \frac{\partial \mathcal{V}}{\partial \mathbf{x}}\dot{\mathbf{x}} = \frac{m}{2}\dot{\mathbf{x}}^T\ddot{\mathbf{x}} - \nabla U\dot{\mathbf{x}}. \tag{3.19}$$

and substituting the system dynamics (3.16) and the equilibrium $\mathbf{x} = \mathbf{x}_d$ into $\dot{\mathcal{V}}$ leads

$$\dot{\mathcal{V}} = \dot{\mathbf{x}}^T\dot{\mathbf{x}} \times \mathbf{B} - k_d\dot{\mathbf{x}}^T\dot{\mathbf{x}}. \tag{3.20}$$

Furthermore, the specific construction $\dot{\mathbf{x}}(\dot{\mathbf{x}} \times \mathbf{B}) = 0$ can be used to simplify the equation significantly. Therefore, one may write

$$\dot{\mathcal{V}} = -k_d\dot{\mathbf{x}}^T\dot{\mathbf{x}} \leq 0. \tag{3.21}$$

This implies that $\mathbf{x} = \mathbf{x}_d, \dot{\mathbf{x}} = 0$ is globally asymptotically stable based on the LaSalle Criterion. Thus, the CF method has no local minimum and furthermore, it ensures collision avoidance due to the inverse square law of the distance to the obstacle.

However, for discrete calculation steps, large objects, and a constant attractor, the point mass can still be pulled into an object. This effect occurs since CFs generate only forces that are orthogonal to the B-field. By altering the attracting and damping force, collisions with an infinitely long wall or a deadlock caused of discretization can be avoided. The main influence that determines such behavior is the definition of the current direction at runtime. Especially in the case of a dynamic environment this task becomes non-trivial. A novel simple and effective approach to define particular current elements on obstacle surfaces is described in Sec. 3.3.2.

Next, a velocity dependent Optical Flow approach is described.

3.1.5 Optical Flow

Optical Flow is the contrast change in a visual scene caused by the relative motion between an observer and the scene. In a 2D time variable case a pixel at location $p(t)$ with intensity $I(p(t))$ will have moved by $\Delta p(\Delta t)$ between two image frames. Therefore, the new image frame is described by

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \text{H.O.T.} \tag{3.22}$$

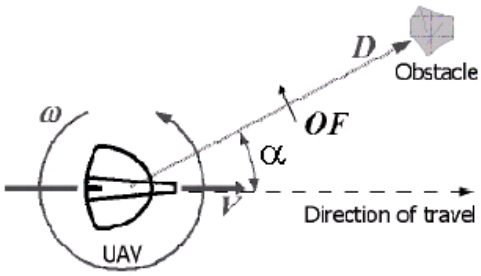


Figure 3.7: Use principle of 1D Optical Flow for collision avoidance.

Assuming that the movement is small and the intensity of the images is constant, we may conclude (by first order Taylor series expansion)

$$\frac{\partial I}{\partial t}\Delta t + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y = 0 \tag{3.23}$$

or alternatively

$$\frac{\partial I}{\partial t} + \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} = 0. \tag{3.24}$$

$\frac{dx}{dt} = v_x$ and $\frac{dy}{dt} = v_y$ are the velocity components and $\frac{\partial I}{\partial x} = I_x, \frac{\partial I}{\partial y} = I_y$ are the pixel intensity gradients. We may now rewrite

$$I_x v_x + I_y v_y = \nabla \mathbf{I}^T \cdot \mathbf{v} = -I_t \ . \tag{3.25}$$

This is an equation with two unknown and cannot be solved as such [5]. To find a solution for this problem methods as the Lucas-Kanade method [41] or Horn-Schunck scheme [24] introduce additional constraints to calculate the optical flow. Lucas and Kanade e.g. simplify the equation to

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} = - \begin{pmatrix} \sum I_t I_x \\ \sum I_t I_y \end{pmatrix} \tag{3.26}$$

and create hereby a fully determined system. For collision avoidance optical flow is often used to adapt the motion direction of the AO as shown in Fig. 3.7. In [18] the 1D Optical Flow is utilized to align the yaw-axis of an UAV as a function of the absolute obstacle distance D, the relative angle α, and the absolute velocity V. The respective angular velocity equation becomes

$$\omega = \frac{V}{D}\sin(\alpha) - OF \ . \tag{3.27}$$

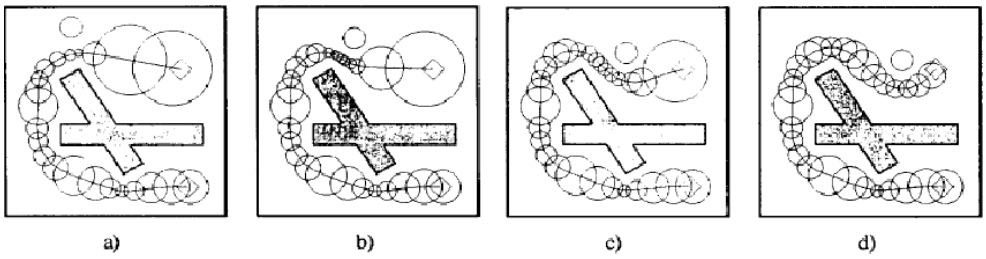


Figure 3.8: Elastic Strips - connecting path planning and control.

Algorithms based on Optical Flow are particularly efficient for their use on direct sensor input and are often implemented in flying vehicles.

Next, we describe an approach that unifies reactive schemes and planning approach: the Elastic Strips framework.

3.1.6 Elastic Strips

Elastic Strips is a method that combines reactive and planning motion generation. Its general idea is to initially plan a collision free joint path with an offline planner and to modify this afterwards during runtime. This is performed by force based algorithms that implement an elastic behavior between generated via points. These so called elastic strips generate a path by an artificial repulsive force $\mathbf{F}_r$ and contracting forces $\mathbf{F}_c$ [8].

$$p_{new,i,j} = p_{old,i,j} + \alpha(\mathbf{F}_r + \mathbf{F}_c) \tag{3.28}$$

Each strip j represents a joint of the robot with discrete point $\mathbf{p}_i$ surrounded by a bubble that covers the strip. The mostly used scheme for force generation is the Potential Fields method. Thus, every bubble acts similar to an avoiding point mass in the Potential Fields framework, see Sec. 3.1.1. Repulsive forces can therefore be written as

$$\mathbf{F}_r = k_r(d_0 - \|\mathbf{d}\|)\frac{\mathbf{d}}{\|\mathbf{d}\|} \qquad \|\mathbf{d}\| < d_0, \tag{3.29}$$

where k_r is the repulsive force factor, d_0 is a scalar constant for the influence sphere, and $\mathbf{d}$ is the vector between point $\mathbf{p}_i$ and the closest point on the obstacle. The contraction forces acting against the repulsive obstacle forces are defined as

$$\mathbf{F}_{c,i,j} = k_c \left(\frac{d_{i-1,j}}{d_{i-1,j} + d_{i,j}} (\mathbf{p}_{i+1,j} - \mathbf{p}_{i-1,j}) - (\mathbf{p}_{i,j} - \mathbf{p}_{i-1,j}) \right), \tag{3.30}$$

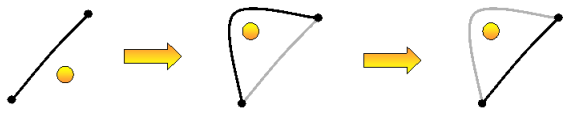


Figure 3.9: Local re-planning in the Elastic Strips framework.

where k_c is the contracting factor and $d_{i,j} = \|\mathbf{p}_{i-1,j} - \mathbf{p}_{i,j}\|$ is the distance in the initial, unmodified trajectory. The contraction forces act between adjoining bubble elements $i-1$, i, and $i+1$ as virtual springs and induce a minimized path length. Extensions of the original Elastic Strips framework include different contraction force calculations and local re-planning if obstacles cross the originally generated path, see Fig. 3.9

Next, we describe planning algorithms, beginning with reactive planning schemes.

3.1.7 Reactive Planning

Reactive planners require a basic representation unit and rules to compose these units into plans. Building a reactive planner imposes to create an intelligent decision structure. In general, reactive planning can be realized with many approaches. The most common ones are condition-action rules [9], finite state machines [28], fuzzy approaches [50], and connectionist approaches [42]. In the following we describe finite state machines and steering by reactive algorithms only, as they are directly related to this thesis.

Finite State Machines

Finite state machine (FSM) describe the system behavior by a set of states and condition activated transitions between these states. In FSMs there is only a single state activated at the same time and only its interrelated transitions need to be evaluated. If a transition condition is true a new state is activated. A self activation of the current state is also possible by a respective transition.

There are several action types possibly executed based on the particular situation. When entering a state, the entry action is executed, when exiting the state, the exit action is executed. Depending on the current state and its associated conditions the input action is executed. While changing the state the dependent transition action is executed.

Those actions can be atomic or rather complex scripts that in turn contain a list of atomic actions. Depending on the complexity it is also common to generate highly hierarchical structures. In such a new automaton every state may contain substates and only the states at the atomic level are associated with a script or another atomic action.

Automatons that are practically in use are often mixed models or further developed concepts of the standard Moore or Mealey automata's.

Steering by Reactive Algorithms

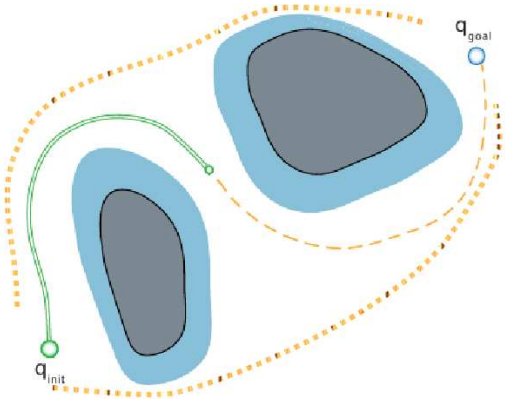


Figure 3.10: Reactive planning algorithms.

The group of steering reactive planning algorithms utilize such algorithms as the ones described in Sec. 3.1.1 to 3.1.5 for generating several trajectories simultaneously and then select the best among them, Fig. 3.10. Typical approaches for solution optimization are

- scattering of the start and end point to draw several configurations and
- variation of algorithm parameters.

Such an optimization process significantly reduces local minima.

Next, we discuss the class of probabilistic planners, which gained enormous popularity in the robotics community during the last decade.

3.1.8 Probabilistic Planning

Probabilistic planning searches the configuration space C_{space} by means of an algorithmic search that draws sample configurations from C_{space} in a random

manner. There are two large classes of probabilistic planners, which approach the problem quite differently: Probabilistic Roadmaps (PRM)s and Rapidly Exploring Random Trees (RRTs).

Probabilistic Roadmaps

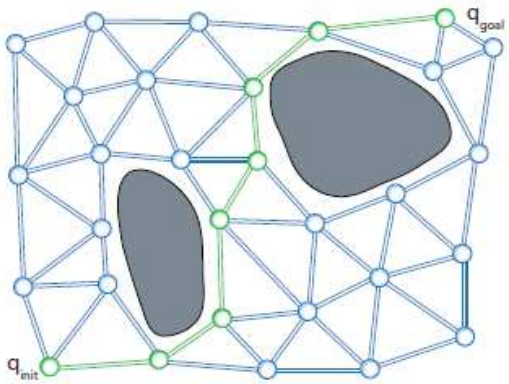


Figure 3.11: Propabalistic Roadmaps.

Probabilistic Roadmaps consist of two elementary phases. First, the creation of a topological graph named roadmap is carried out. Then a graph search in this map for a connectible path between q_{init} and q_{goal} [36] is performed.

The creation of the roadmap is also called the learning phase, where random collision free configurations are drawn and connected via a local planner. In this creation process the nearest neighbors in each case are linked to each other so that a dense network with good topological covering propoerties is created. The complete network is a collision free map in which the configuration space can be traversed in particular ways.

The second phase is the exploration phase where the connection of start configuration q_{init} and target configuration q_{goal} is performed. If this is successful, the individual nodes are combined into a path P. Then, P is converted via a smoothening step into a feasible path, see Fig. 3.11.

Once created, roadmaps can still be used for further path planning as long as the scene remains static. If after repetitive unsuccessful searches no results is found, it is possible to extend the roadmap accordingly. Commonly, these algorithms favor undiscovered areas. In case of a slowly changing environment a successive collision check may be performed while moving.

Thus, smaller scene changes can be treated, however, a dynamic environment increase the calculation time exponentially.

Probabilistic Roadmaps are ideally suited for industrial scenarios where the same task has to be performed several times. This s especially due to the generation of the roadmap, which detailed structure is the computationally largest part of the algorithm. However, after the initial generation the search can be performed at relatively high speed. Nonetheless, due to the necessary construction of the roadmap, this method is not well suited for dynamic environments necessitating its frequent re-computation.

Next, we discuss the class of Rapidly Exploring Random Trees (RRTs).

Rapidly Exploring Random Trees

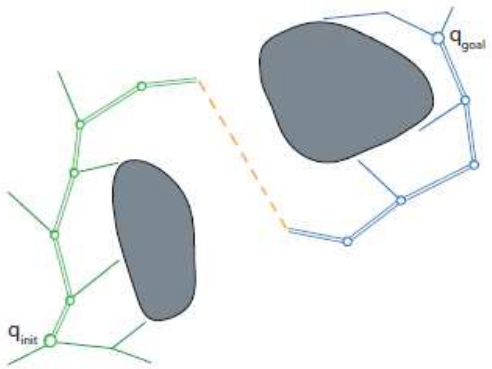


Figure 3.12: Bidirectional Rapidly Exploring Random Trees.

Rapidly Exploring Random Trees is a large family of algorithms for probabalistic exploration of C_{space}. The idea is to incrementally construct a search tree that gradually improves its resolution. On the long term the tree denseley covers the space. The method has certain similarities with space filling curves. However, instead of generating one long path it generates shorter paths that are organized in a tree type datastructure. In general, a dense sequence of samples is used for incrementally constructing the tree. If this sequence is random, the resulting tree is called a rapidly exploring random tree (RRT) [33] [35].

The characteristic properties of RRTs are

- target-oriented search,
- rejection of individuals with poor quality during search,

- and a possible bi-directional search with connectivity review.

Therfore, RRTs are well suited for unique requests as they are target oriented and effective. An uncharted space can be explored quickly, whereby a feasible path is found contemporary. Unlike PRMs, RRTs do not process an invalid network after a manipulation of objects. With a second search tree a quick solution for target configurations in remote areas can be found. Despite its unquestionable advantages RRTs are currently not applicable for dynamic scenarios as it usually takes still several seconds calculation time before finding a respective solution.

3.2 Comparison and Pre-selection

For selecting the appropriate methods for further investigation we discuss now the characteristic properties of the aforementioned algorithms Sec. 3.1. Especially calculation time and collision avoidance performance are of significance.

Local Minima, Task Relation and Applicability

The main characteristics of the discussed algorithms are listed in Tab. 3.1. We differentiate by means of task fulfillment, collision avoidance, and the ability to provide output values for desired force-torque, velocity, or position. This intends to divide the algorithms in the sense that some focus on reaching a target pose, as e.g. HPFs and CFs and other primarily avoid collisions as PFs and its derivates (the group of PF adaptations as e.g. HNPFs, adaptations for moving obstacles, or cancellation of local minima for spherical obstacles [31, 32]). OFs are is particularly endangered to collide with

	Algorithms	Local minima	Task fulfillment	Affected quantities	Calculation time	Parallelizability
location dependent	**PF**	yes	secondary	f, τ, $\dot{x}$, x	very low - low	high
	HNPF	yes	secondary	f, τ, $\dot{x}$, x	very low - low	high
distance& motion dependent	**HPF**	no	main focus	f, τ (?), $\dot{x}$, x	low	(?)
	CF	no	main focus	f, $\dot{x}$, x	low	high
	OF	yes (symmetry)	layout dependent	f, $\dot{x}$	very low	high
	Elastic Strips	no	main focus	x	medium	-
	Reactive planner	no	given	$\dot{x}$, x	medium - high	low
	Probab. planner	no	given	x	high	medium

Table 3.1: Comparison of methods by characteristic behavior.

symmetrical obstacles for which the OF may become zero. PFs, HPFs, and CFs are adaptable such that they may provide output reference values for low-level controllers by means of desired force-torque, velocity, and position values. Planners in the contrary, provide exclusively position trajectories.

Calculation Time and Parallelizability

Another very important aspect of algorithms is the calculation time on currently available hardware platforms and also their potential parallelizability. OF algorithms, e.g., already use direct sensor input and can be highly integrated into appropriate hardware as e.g. FPGAs. Reactive algorithms as described in Sec. 3.1.1 to 3.1.5 can be calculated at very high control rates in the range of ≈ 1 ms (very low - low). These algorithms are highly parallelizable and a further advance is that the combination of multiple algorithms leads only to a minimal increase in calculation effort. Using the latest evolution in graphics programming as CUDA, OpenCL, or Cg (C for graphic), which use different types of pipeline restructuring or shader programming, it can be advantageous to implement final applications of these algorithms on graphics processors and use their enormous calculation power (this is measured in giga floating point operation per second (GFLOPS)). Currently available high-end CPUs theoretically provide 10 (Pentium 4) up to 80 GFLOPS (Core i7), while graphic boards of Nvidia and ATI 1000 are able to process 1300 GFLOPS. Further insight on the resulting calculation effort of selected algorithms and the concrete parallelizability is given in Sec. 3.3.2.

Planners in turn are often not strongly parallelizable and mostly characterized by serial decision process. In this process varying data access is necessary and the process itself is not deterministic. Therefore, these algorithms especially benefit from calculations on a CPU architecture.

Algorithm Selection

For further consideration, algorithms with different properties are selected to compare them in simulation and then perform another selection for testing them experimentally. For low-level reactive motion generation planning algorithms are not fast enough and are therefore excluded from this analysis. Elastic Strips are also not considered anymore, as the require a global planner for initialization. Also their calculation time is expected to be sufficiently low for not running in the inner most control loop. Therefore, we

choose Potential Fields and Human Navigation Potential Fields as position based schemes and Optical Flow and Circular Fields as motion dependent algorithms. Since Harmonic Potential Fields are expected to have similar properties and capabilities as Circular Fields, we leave them for future research.

Reactive planners are not fully excluded from the analysis because they still can be used for implementation on a logical level as will be shown in Sec. 6.3.

In the following we discuss the extensions we have made to adapt the selected algorithms so they generate the desired behavior.

3.3 Algorithmic Extensions

During the analysis of the chosen algorithms some deficits became apparent, which led to significant improvements of the schemes. Especially for the CFs method we carried out several extensions and also developed a hybrid motion generation approach, composed of CFs and PFs. Before discussing these extensions we first introduce the concept of our robot hull design, which is the geometric abstraction we utilize for the collision avoidance schemes in 6D.

3.3.1 Robot Hull Design

In this thesis a robot is formally separated into several avoiding objects (AOs) that create a partitioned geometrical representation of the robot. This robot representation is organized into several geometrical hulls that cover

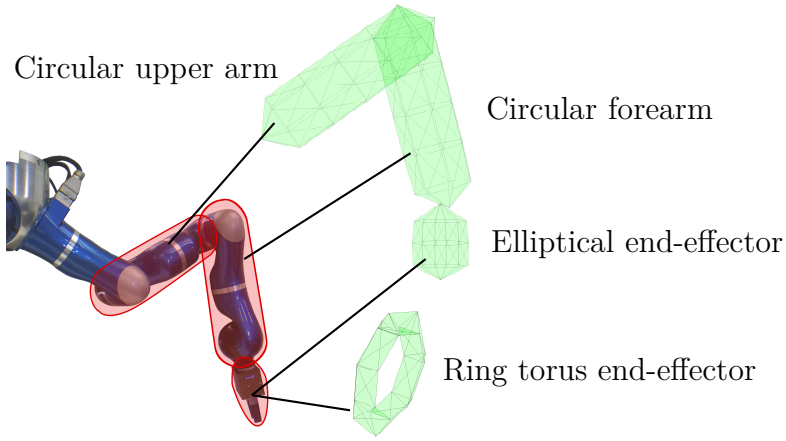


Figure 3.13: The implemented, geometrical robot hull, for the LWR-III.

the robot with surface elements and endow it with a volume representation for all actively movable parts. Thereby, we can generate virtual forces and moments on each separate robot segment. The current segments are chosen such that they represent the end-effector, the forearm, and the upper arm, see Fig. 3.13. The end-effector can be freely chosen as point mass, ellipsoid bar hull, or the special ring torus (used later for the tactile exploration in Sec. 6.3). The upper and forearm are equally defined circular bar objects. All objects are described in more detail in Sec. 5.1.3. In the future the current implementations shall be substituted by automatically generated hulls as the ones in [15].

Next, the developed extensions of the CF approach are described.

3.3.2 Circular Field Enhancement

As described above, the goal convergence for the CF point mass approach is proven for 3D motion. In this thesis the model is extended to 3D avoiding bodies moving in 6D. To achieve appropriate motion generation we extended the algorithm significantly and present its most important enhancements. Furthermore, we outline an approach to reduce the calculation load.

The first novel contribution is the CF current definition as described in the following.

Adapting the Surface Current Rotation Vector

In the publication of the CF approach [53] two schemes for current definition where given. They generate independent current elements for every surface and therefore, induce especially for the 3D case oscillating behavior, see [53] Fig. 6-7. Due to various problems in applying these algorithms already in simulation, we developed an alternative way of defining the current directions, which was published in [20]. There, we define the current of all CF obstacle surfaces interrelated to each other. The used approach is defined on the base of the vector from the actual AO position $\mathbf{x}$ to the desired position $\mathbf{x}_d$, the goal vector $\mathbf{b}$ and the center of mass of the respective OO $\mathbf{m}_{og,j}$, see Fig. 3.14.

The local circular field $\mathbf{B}_{j,i}$ of each surface element, acting on the virtual particle k is defined as

$$\mathbf{B}_i := I_K \frac{(\mathbf{n}_i \times \mathbf{r}_j) \times \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|}}{l_i^2} \, \mathrm{d}a_i, \tag{3.31}$$

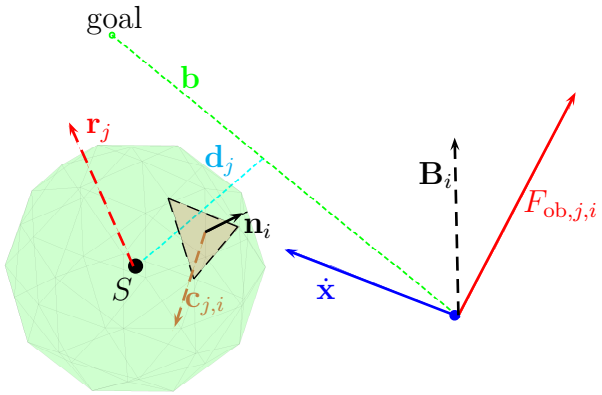


Figure 3.14: Principle of the current definition for Circular Fields.

where I_K is the virtual current, $\mathbf{n}_i$ is the normal of surface element i, $l_i = \|\mathbf{x}-\mathbf{x}_{\mathbf{n}_i}\|$ is the distance of the current configuration $\mathbf{x}$ to the surface element, and $\mathbf{x}_{\mathbf{n}_i}$ is the position of the surface element. In this equation the surface current of (3.15) is defined as $\mathbf{c}_{j,i} = \mathbf{n}_i \times \mathbf{r}_j$.

$\mathbf{r}_j$ is the field rotation vector for an obstacle, which is defined as

$$\mathbf{r}_j := \frac{\mathbf{d}_j \times \mathbf{b}}{\|\mathbf{d}_j \times \mathbf{b}\|}, \tag{3.32}$$

with $\mathbf{d}_j$ being the shortest distance between the center of mass of the OO $\mathbf{m}_{og,j}$ and the goal vector $\mathbf{b}$:

$$\mathbf{d}_j = \mathbf{x} + \mathbf{b}\frac{(\mathbf{m}_{og,j} - \mathbf{x}) \cdot \mathbf{b}}{||\mathbf{b}||^2} - \mathbf{m}_{og,j} \tag{3.33}$$

With the given definition the current continues around the object and therefore oscillating robot behavior is avoided as long as the damping and attractor are chosen according to Sec. 3.1.4 or as in the following.

Adapting Attractor

If the attractor is defined according most potential field approaches as

$$\mathbf{F}_a = -k_a(\mathbf{x} - \mathbf{x}_d), \tag{3.34}$$

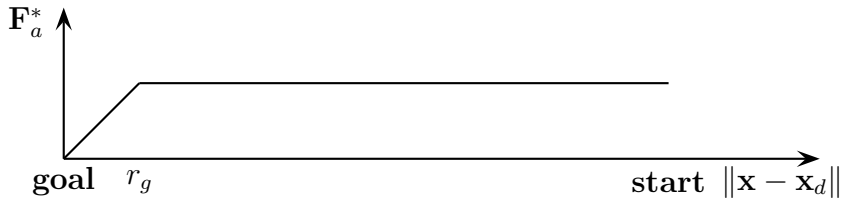


Figure 3.15: The attractor profile used for simulation.

this may lead to increasing holding influence of the obstacle CF during goal converging motion. This means that the AO circulates e.g. 1 or 3 times around the obstacle, however, converging afterwards towards the goal. To reduce this behavior in simulation we choose the attractor to be constant until being close to the goal. When being sufficiently close to the goal (r_g is the range of the non-constant potential field attractor) the influence of the attractor is reduced towards zero until reaching the goal, see Fig. 3.15. Next, we describe the velocity angle adaptation for CFs.

Velocity Angle Adaptation

The chosen current definition leads to diverting forces in front of the obstacle and forwarding forces behind it. Changing the direction according to the obstacle rotation vector $\mathbf{r}_j$ about the angle ψ

$$\dot{\mathbf{x}}^* = \mathbf{R}\left(\frac{\mathbf{r}_j}{\|\mathbf{r}_j\|}\psi\right)\dot{\mathbf{x}} \tag{3.35}$$

leads to

- earlier deviation (also if being parallel to an infinite wall)
- and to reducing the CF influence after passing by the obstacle (when not being that deep in the CF anymore),

Perception Shaping

In this thesis we implemented various perception shaping strategies in the calculation of the CF-forces. The first two methods, described hereafter, are used in general and are position dependent. The third method reduce the OOs influence when objects are passed or are not in the current motion direction.

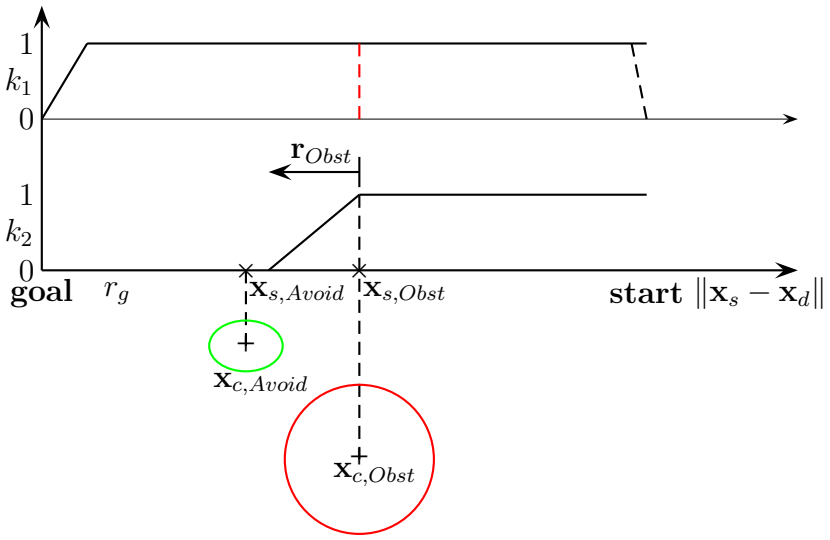


Figure 3.16: Static CF influence shaping between start and goal.

The normal has to be in the influencing range. This limitation on perception range is the first step to determine the process of environmental force calculation for randomly sized environments. This is an admissible simplification because even global calculations are generally dominated by the closest surface or line elements.

The distance surface normal scalar product has to be positive $0 < (\mathbf{x} - \mathbf{x}_{n_i}) \cdot \mathbf{n}_{j,i}$ to ensure that the obstacle surface can be observed from the point the force is acting on. Thereby, influences and problems by fine objects are reduced, stronger deviation can be generated, and less surfaces need to be calculated.

Statical CF shaping depending on the distance between the avoiding and OO is used to reduce the obstacle influence by a linear decreasing function if the obstacle is farer from the goal than the AO. The according profile is depicted in Fig. 3.16. By the multiplication with the scale factors k_1 and k_2 the final CF scale factor $k_{s,CF} = k_1 k_2$ is obtained. This scaling is a robust method to avoid oscillating behavior with attractors on other levels of robot control.

Calculation Load Reduction

The construction of obstacle and AOs based on surfaces results in an exponential increasing effort in calculation effort for the equations described in Sec. 3.1.4. Thus, it is of large interest to simplify the algorithm without loosing performance and analyze to introduce parallellizability. For AO n and OO j that are created from AO surfaces k and OO surfaces i the force equation includes two sum formulations.

$$\mathbf{F}_{obj,j,n} = \sum_k \sum_i (\dot{\mathbf{x}}_{k,i,rel} \times \mathbf{B}_{i,k}) \tag{3.36}$$

In simulation experiments it turned out to be disadvantageous to include the relative velocity $\dot{\mathbf{x}}_{n,k}$ of surfaces k to the body center point of the object n. Thus, we consider the relative velocity to the body center point approximately to be zero and therefore negligible. The velocity of every avoiding surface element is now the relative velocity $\dot{\mathbf{x}}_{n,\text{relative}}$ of AOs n to OOs j. Because of this simplification it is possible two reduce the to sum formulations as well. This leads to

$$\mathbf{F}_{obj,j,n} = \dot{\mathbf{x}}_{\text{rel}} \times \sum_k \sum_i \mathbf{B}_{i,k}, \tag{3.37}$$

where j is the actual OO and n is the AO the force acts on. In the following, we define $\dot{\mathbf{x}} = \dot{\mathbf{x}}_{rel}$. Furthermore, the equation of $\mathbf{B_i}$ is now

$$\mathbf{B}_{i,k} := I_K \frac{(\mathbf{n}_{j,i} \times \mathbf{r}_j) \times \frac{\dot{\mathbf{x}}}{||\dot{\mathbf{x}}||}}{l_i^2} \, \mathrm{d}a_i \, \mathrm{d}a_k, \quad \text{with} \quad \mathbf{c}_{j,i} = \mathbf{n}_i \times \mathbf{r}_j. \tag{3.38}$$

In addition, we consider that the current $\mathbf{c}_{j,i}$ is defined to be constant or calculated ahead. If we also regard that the objects are constructed with approximately equally sized surfaces, the calculation effort reduces further and the equation is simplified to

$$\mathbf{B}_{i,k} := I_K \frac{\mathbf{c}_{j,i} \times \frac{\dot{\mathbf{x}}}{||\dot{\mathbf{x}}||}}{l_i^2} \, \mathrm{d}a^2. \tag{3.39}$$

A further simplification can be achieved by including individual obstacle surface size factor in the individual current direction vector of every surface. With this it is possible to change the sum formulation as follows.

$$\sum_k \sum_i \mathbf{B}_{i,k} = I_K \, \mathrm{d}a^2 \sum_k \sum_i \frac{\mathbf{c}_{j,i}}{l_{i,k}^2} \times \frac{\dot{\mathbf{x}}}{||\dot{\mathbf{x}}||} \tag{3.40}$$

$$= -k_{B_i} \frac{\dot{\mathbf{x}}}{||\dot{\mathbf{x}}||} \times \sum_k \sum_i \frac{\mathbf{c}_{j,i}}{l_{i,k}^2} \quad \text{with} \quad k_{B_i} = I_K \, \mathrm{d}a^2 \tag{3.41}$$

Thus, the object force can be written as

$$\mathbf{F}_{obj,j,n} = -k_{B_i} \left(\dot{\mathbf{x}} \times \frac{\dot{\mathbf{x}}}{||\dot{\mathbf{x}}||} \times \sum_k \sum_i \frac{\mathbf{c}_{j,i}}{l_{i,k}^2} \right). \tag{3.42}$$

Now, the calculation effort is already significantly smaller and could be further reduced with a voxel space approximation for $l_{i,k}^2$. This leads to determining the distance $l_{i,k}$ by the voxel grid. Another result of this simplification is that the algorithm could be calculated highly in parallel. Most operations could be carried out e.g. 100 obstacle surfaces and 100 AO surfaces up to 100.000 times in parallel. Furthermore, a parallelization of the parts of the avoiding robot is advantageous for multiple avoiding objects (e.g. several objects attached to the different robot links).

Additional Calculation for PF In addition to the CFs we also calculate the corresponding potential fields for each obstacle surface acting on every avoiding surface. Therefore, the calculation of CF forces done before can be used for the generation of CF forces at the same time. If only the potential field forces need to be calculated for every avoiding surface, the equation

$$\mathbf{F}_{n,k} = k_r \sum_i \frac{\mathbf{x} - \mathbf{x}_{n_i}}{\|\mathbf{x} - \mathbf{x}_{n_i}\|^2} \tag{3.43}$$

has to be solved. This means that the minimal calculation effort $\forall i$ is 8 additions, 3 multiplications, and 3 divisions. This leads to 14 floating point operations (FLOs) in total. Using the data $\mathbf{dx}_i = \mathbf{x} - \mathbf{x}_{n_i}$ and $l_i^2 = \|\mathbf{x} - \mathbf{x}_{n_i}\|^2$ already calculated for the CF forces, the remaining calculations are

$$\mathbf{F}_{n,k} = k_r \sum_i \frac{\mathbf{dx}_i}{l_i^2}. \tag{3.44}$$

In other words, 14 FLOs are reduced to 6 FLOs, or 3 additions and 3 divisions, so a reduction of calculation load on 43 %.

Example Calculation Load The already started assignment of FLOs per calculation period can be extend to an environment with 10 obstacles and 1 avoiding object, of which each is represented by 100 surfaces to estimate the necessary calculation power. As an example we assume

$$\mathbf{F}_{obj,j,n} = -k_{B_i} (\dot{\mathbf{x}} \times \frac{\dot{\mathbf{x}}}{||\dot{\mathbf{x}}||} \times \sum_k \sum_i \frac{\mathbf{c}_{j,i}}{l_{i,k}^2} \, \mathrm{d}a_i \, \mathrm{d}a_k). \tag{3.45}$$

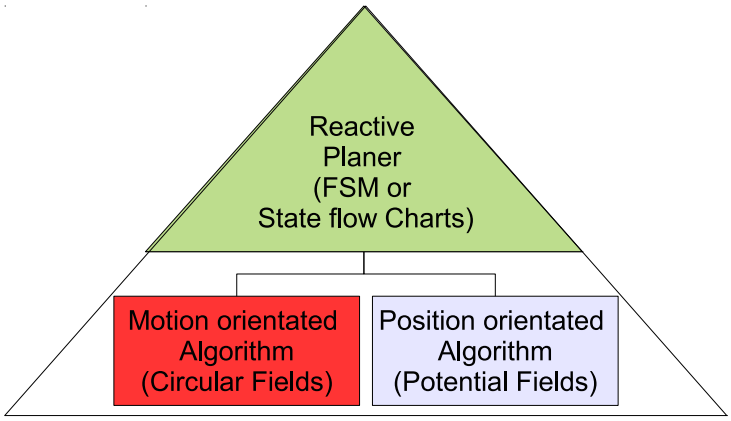


Figure 3.17: Hierarchical reactive collision avoidance.

with individual surface sizes to achieve a representative result. In front of the sum formulations there are only the operations "scalar multiplied with a vector", two "cross products", and a "vector normalization". This results in total 33 FLOs. However, for the given example the calculations needs to be performed in this particular execution order and is not parallelizable. In contrast, the two sum formulations are fully parallelizable 100.000 times. $l_{i,k}^2 = \|\mathbf{x}_k - \mathbf{x}_{n_i}\|^2$ results in 8 FLOs, multiplied with $\mathrm{d}a_i$ $\mathrm{d}a_k$ (again 2 FLOs), and $\mathbf{c}_{j,i}$ divided by the result (3 more FLOs). The general aim is to reach an overall rate of 1 kHz. The additional calculations of the previously described PF forces and this particular CF force calculation require $600.000 + 1,300.000$ FLOPS ≈ 2 GFLOPS. Thereby a real-time calculation is possible on a CPU (10 up to 80 GFLOPS).

3.3.3 Hybrid Approaches

As described in Sec. 1 one of the main challenges of this thesis is the task related motion generation in a nearly collision free manner. To fulfill this specification it can be advantageous to combine algorithms. Different combinations are of course possible and a hierarchical structure to be developed. This can be represented by the pyramid structure depicted in Fig. 3.17.

As motion orientated algorithm also Optical Flow and especially Harmonic Potential Fields can be considered. As position orientated algorithms other Potential Fields derivates are potentially advantageous. In this thesis, however, we chose to use CFs for translation motion and PFs for rotation, i.e. to generate moments in the 6D simulations and experiments, see Sec. 6.3.

Reasons to do so are:

- Attracting (diverting in front of the obstacle center) and repulsive forces (forwarding behind of obstacle center) while passing objects lead to unwanted rotations for CFs.

- A stability proof is not yet available for the rotational energy part (B-field injects rotational energy with our extensions).

- The approach is advantageous to decouple the translation and rotation motion clearly from each other.

- Potential fields lead to repulsive behavior for rotations.

- The chosen solution is a robust base for future extensions, as an attractor for the orientation and optimization of the current $\mathbf{c}_{j,i}$ definition.

The force equations that are used to generate desired virtual environment forces or for dynamic simulation, desired velocity, and position data can be written as:

$$\mathbf{f}_{obst,n} = \sum_k \mathbf{f}_{CF,n,k} \tag{3.46}$$

$$\mathbf{m}_{obst,n} = \sum_k (\mathbf{x}_{n,n_k} - \mathbf{x}_{ap,n}) \times \mathbf{f}_{PF,n,k} \tag{3.47}$$

$$\mathbf{F}_{obst,n} = [\mathbf{f}_{obst,n} \quad \mathbf{m}_{obst,n}]^T, \tag{3.48}$$

where $\mathbf{f}_{CF,n,k}$ is the Circular Field force of the AO n on its k^{th} surface element, $\mathbf{f}_{PF,n,k}$ is the Potential Field force that results by the lever arm from $\mathbf{x}_{ap,n}$ the position of the center of mass (or anchor point) to the force corresponding normal position $\mathbf{x}_{n,n_k}$. In order to solve the problem that an OO j is initialized in the AO n, or that the OO and AO surfaces are crossing each other, we also use PFs in translational motion when such cases occur. Therefore, CFs loose their particular influence and the role of PFs rise in a transition zone until exclusively PFs are used if the AO particle i is penetrating the

virtual OO. The gains of CFs k_{CF} and PFs k_{PF} are defined as follows.

$$k_{\mathrm{ratio}} = \frac{\|\mathbf{x} - \mathbf{x}_{c,j}\|}{\|\mathbf{x}_{c,j} - \mathbf{x}_{j,n_i}\|} \tag{3.49}$$

$$k_{\mathrm{CF}} = \begin{cases} 1 & \text{, for } (1+\delta_{\mathrm{ratio}}) < k_{\mathrm{ratio}} \\ \frac{k_{\mathrm{ratio}}-1}{\delta_{\mathrm{ratio}}} & \text{, for } (1+\delta_{\mathrm{ratio}}) > k_{\mathrm{ratio}} \\ 0 & \text{, for } 1 < k_{\mathrm{ratio}} \end{cases} \tag{3.50}$$

$$k_{\mathrm{PF}} = 1 - k_{\mathrm{CF}}, \tag{3.51}$$

where k_{ratio} is the ratio of the distance of center of OO $\mathbf{x}_{c,j}$ to the AO particle and the distance from center of OO to its normal position $\mathbf{x}_{j,n_i}$. δ_{ratio} defines the size of transition zone. This behavior is especially advantageous when using the CFs shaping and an object approaches from behind.

4 Structured Analysis

In this chapter the structured analysis of the DLR Co-Worker system is conducted exemplarily to support the determination of an appropriate solution approach for the given collision avoidance problem. This formal tool allows to structure a system such that it becomes manageable and its internal coherence is transparent. We follow the formal procedure described in [29].

To get a general overview of the Co-Worker scenario this chapter starts with a description of the hardware available setup followed by a description of its software architecture. Using this knowledge the user requirements of the task are defined subsequently and then taken into account for the system analysis. Furthermore, we generate new functionalities and integrate them into the context diagrams. Then, the data flow diagrams and the architecture diagram are developed. Taking this into account specifications of the main and sub-functionalities as well as a concretion of purpose is done. Finally, the system requirements are combined.

4.1 Co-Worker Scenario

The Co-Worker scenario is a non-mobile multi-robot platform for developing and testing methods for physical Human-Robot Interaction (pHRI), see Fig. 4.1. The setup is based on LWR-III manipulators and is equipped with various exteroceptive sensors. The first version of the system consisted of a single manipulator (see Fig. 4.1 left), while the newest setup combines three LWR-IIIs (see Fig. 4.1 right) into a single demonstrator platform. The exteroceptive sensing capabilities consist of the DLR 3D-Modeler [57], Time-of-Flight Cameras mounted on two manipulators, an external infrared tracking system, and a camera in the center of the LWR-III Adapter. Both systems are used for validation in this thesis and as testbeds for the simulation and experimental results. The system offers options of cooperation scenarios by means of

1. multi-robot interaction
2. (mutli-)robot-human interaction

In the following, we shortly describe the sensing modalities in more detail.

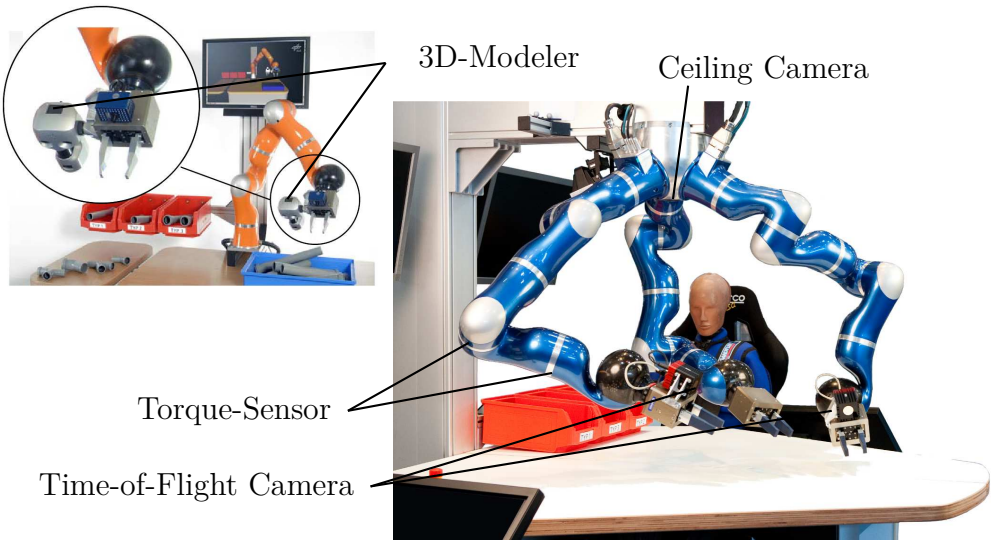


Figure 4.1: The DLR Co-Worker scenario.

Proprioceptive Perception Internal sensors of the Co-Worker setup are the position and torque sensors of the LWR-IIIs. The LWR-III measures motor positions by digital Hall effect sensors and its link side positions by encoders. Furthermore, it measures torque values with special profiles and strain gauge based sensors in every joint. A disturbance observer uses the measured joint torques and a good model of the dynamics of the robot to estimate external joint torques and uses this to detect collisions with the environment.

Exteroceptive Perception External sensors of the setup are used to create a virtual representation of the environment.

The 3D-Modeler and the Time-of-Flight cameras generate distance information, which can e.g. used to localize and identify objects in the workcell [44]. The Time-of-Flight sensors (SwissRanger SR4000) use an array of infrared LEDs to gather depth images for distances up to 0.5 m and are mounted on top of the end-effectors of two of the three LWR-IIIs.

The infrared tracking system is used to locate passive markers, which can be attached to objects or humans. Therewith, high accurate 6D pose mesuring in large parts of the robots workspace can be achieved. The system with 6 cameras of the type ARTtrack2 is able to cover about two-thirds of the scenario. By derivation of the detected pose, translation and angular velocity of the potential obstacle can be monitored and used to perform

experiments as shown in Sec. 6.3.2

The ceiling camera (see Fig. 4.1) Guppy F146-C from Allied Vision Technology [58] is placed in the robot base above the table to detect objects on the table surface. We use the commercial software HALCON to separate and identify objects by 2D surrounding blobs or polygons as e.g. manipulator elements, human skin, or other types of objects on the table.

To sum up, the DLR Co-Worker scenario with its multiple sensors and actors offers a well equipped platform to develop and test collision avoidance algorithms in complex scenarios.

Next, we discuss the modular and state-based control architecture that is used to control the robots.

4.2 System Architecture

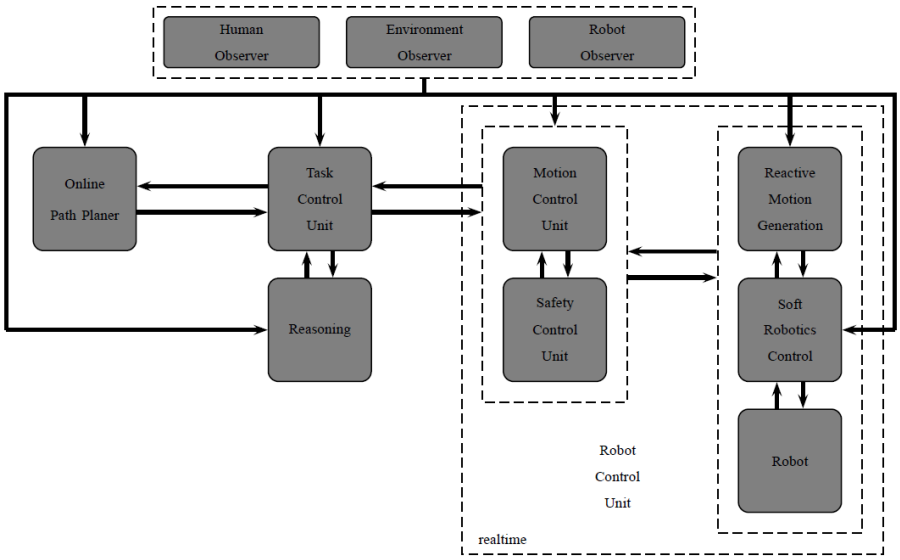


Figure 4.2: Overview of the LWR-III architecture for human-friendly behavior.

To enable efficient robot task and feature development, a clearly structured control architecture is needed, which allows access to safety features, interpolators, controller and sensor data.

The basic structure of this architecture is depicted in Fig. 4.2 and shows the four central entities for robot control:

1. Task control unit (TCU)
2. Robot control unit (RCU)
   a) Safety control unit (SCU)
   b) Motion control unit (MCU)

The first two units serve as the general interface to the robot and communicate with each other via asynchronous protocols. The TCU is the general state based control entity for gathering non-real-time data and providing the correct nominal behavior changes on an abstract level to the RCU. The RCU runs in the same clock rate as the robot and assigns control methods, motion generation schemes, and safety methods. In addition, it interprets and validates the selection of behaviors from the TCU, while preventing incorrect combinations with respect to the current functional mode. The SCU serves as an underlying safety layer below the RCU. It combines all low-level safety behaviors and activates them consistently. The MCU takes care about consistent switching of motion generators and controllers. The particular compatibility between control algorithm and motion generator is encoded in a separate truth table. SCU and MCU are both implemented as state machines.

The RCU is enabled to run at 1 kHz calculation rate. The TCU is a non-real-time system that runs approximately at $20 - 100$ Hz. Both are enable to obtain data from human, environmental and robot observers.

Next, we discuss the user requirements of the new functionality for collision avoidance, the *RCA*.

4.3 User Requirements

Above all, user requirements are based on the existing system and the realization of a flexible application for collision avoidance functionality that is to be developed. The basic user requirements are:

- Integration into existing software: connect the final solution via ArdNet with the RCU and the TCU. Therefore, an interface has to be defined that takes following aspects into account.
- Providing flexible methods for motion generation: different output values as force, velocity, and position should be implemented and their functionality be accessible by commands.

- Consider the potentially complex static environment of the robot: the environment library should include objects, ranging from basic geometries to rather complex ones.

- Registration of dynamic environments: Objects may suddenly occur and move dynamically, therefore a fast and reliable response to changes is required.

The design of the novel functionality should leave freedom of choice by its internal structure.

4.4 System Analysis

The already presented Co-Worker setup from Sec. 4.1 is controlled via the TCU and the RCU. Hence, these are the most important terminators for the new functionality "generate Collision Avoidance Values" (RCA — Reactive Collision Avoidance), see Fig. 4.3.

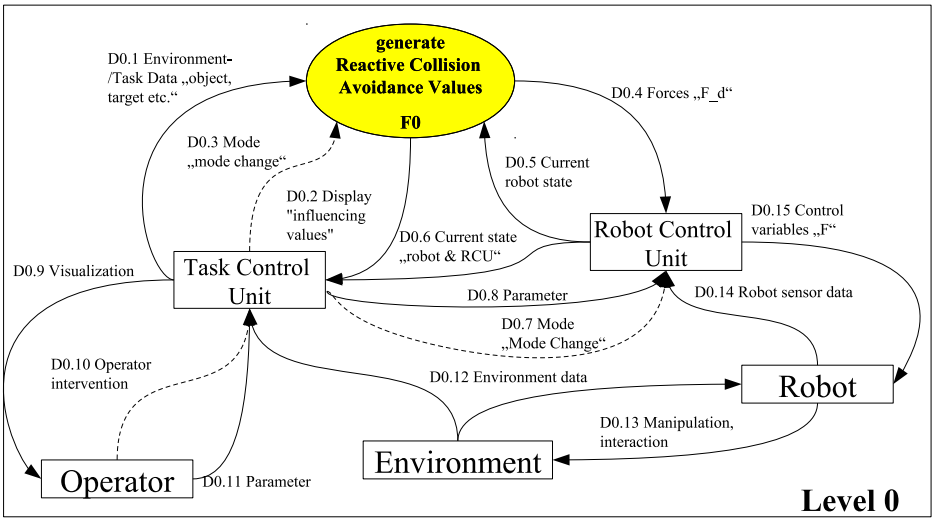


Figure 4.3: Data Context Diagram of the present control software.

The RCA is not directly controllable via the operator. It is an observer of the robot and the perceived (virtual) environment, e.g. obtained by input from the TCU. The robot itself is able to interact with the environment and thereby, changes its own and the TCU sensorial input data.

4.4.1 Task Control Unit specifications

The TCU represent the highest control level in the present system. It

- coordinates the environment perception (human and environment) and forwards it in arranged (conditioned) manner to the RCA,
- performs higher-level decisions for the task fulfillment,
- monitors the pictorial representation of forces, components, and environment,
- and monitors and controls (not real-time) RCU and RCA.

The operator receives visual information about the state of the system which is displayed numerically or with a 3D viewer. He can control the behavior of the system by parameter or binary control. The perception of the environment is performed by several observing systems: the DLR 3D-Modeler, Time-of-Flight Cameras, an external infrared tracking system, and a camera in center of the LWR-III adapter, as described in Sec. 4.1. The TCU controls the RCU by task parameters and modes, as e.g. by sending the desired interpolator or controller type. The interaction with the RCA is described in Sec. 4.4.3.

4.4.2 Robot Control Unit specifications

The RCU represents the contact level between TCU or virtual observer to the integrated controller in the robot. It is also the lowest layer it has to communicated with. It

- consists of multiple dynamical models for control, operation, monitoring, and simulation of the robot (real-time),
- regulates and controls the robot with different methods for motion generation, such as: interpolators, control algorithms, and physical collision monitoring.

Furthermore, there are impedance, velocity, and position controllers implemented. The impedance controllers can react to real or virtual force input. Dynamic models, sensor inputs, and environment data is used to detect collisions. Within a millisecond the RCU reacts to a collision and continues the task autonomously or alternatively consults the TCU in accordance to its current mode.

4.4.3 PSPECs F0 generate_Collision_Avoidance_Values

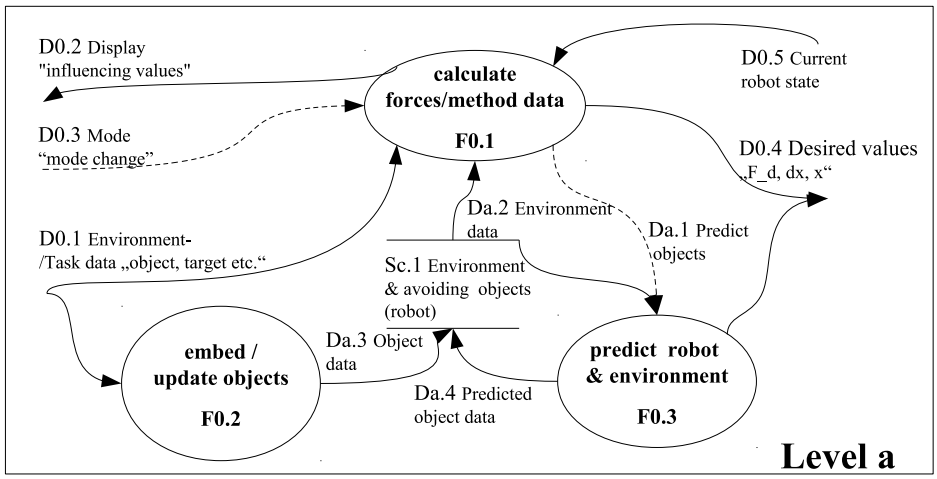


Figure 4.4: Data Flow Diagram of function F0.

We consider the function F0 as a Virtual Environment Observer for Reactive Collision Avoidance (RCA). It

- creates a virtual environment by processing and storing received dynamical 3D Object data,
- uses simple motion prediction algorithms for dynamic obstacle and avoiding 3D objects,
- creates desired virtual forces, velocity, and position values to command and control the robot.

To provide this functionality it is advantageous to split up function F0. The environment is persistently hold by allocated memory. For increasing the clarity of structure separate new sub-functionalities are introduced for calculation of forces or desired values of the chosen method (algorithm) **F0.1**, predict the robot and the environment **F0.3**, and embedding or updating objects **F0.2**. The resulting relations are designed by a data flow diagram, see Fig. 4.4. This illustration allows only a limited view into the functionalities of the RCA. In general, we utilize force based motion generation algorithms. Due to the use of avoiding object dynamics they also provide motion and position data. If the algorithm generates velocity values it can of course be

used also to provide position data by numerical integration. Further function and sequence specifications are given in Chapter 5.

The data flow of function F0 is specified in a data flow dictionary, see Tab. 4.1.

Data Flow	**Description**
D0.1 Environment /Task data	- Target pose by 3×1 vector or 4×4 matrix - Information about the object in the environment as e.g. its kind or size - Pose as a 4×4 matrix - and motion as 3×1 rotation and 3×1 translation vectors
D0.2 Display	- For visualization of the robot influencing values as forces, potentials, or motion
D0.4 Desired values	- The desired values of virtual forces, velocity, or position in order to influence or control the robot
D0.5 Current robot state	- 3 times (for every robot), current pose 4×4 matrix, and Operational space motion 3×2 of the robot
Da.2 Environment data	- Virtual environment data, pose, surface positions, and normals, further specified in Sec. 5.2.1
Da.3 Object data	Object definitions and current state, further explained in Sec. 5.2.1
Da.4 Predicted object data	Changed object data, further explained in Sec. 5.2.1
Control Flow	**Description**
D0.3 Mode	Indicates a mode change
Da.1 Prediction of objects	Signals to predict object motion

Table 4.1: Data Flow Dictionary of function F0.

However, partly the exact structure of allocated memory is still dependent on the finally selected variant of the particular algorithms.

PSPECs F0.1 calculate_forces/motion

For the RCU the RCA provides functions for force, velocity, or position generation. This is depending on the particular method and strategy. It provides the according algorithms for each mode and also relevant visualization data depicting the reactions of the robot for the TCU.

PSPECs F0.2 predict_environment

Predicts the environment for non real-time environment detection and should avoid non-steady values. Furthermore, it provides the possibility to predict individual objects or the entire environment. Thereby, for the case that F0.1 generates e.g. exclusive force data, it is used to provide the respective velocity and position values.

PSPECs F0.3 insert_an/update_object

The RCA provides procedures and functions to integrate objects into the virtual environment and to deleted and/or change their current state and representation. This dependes on the received data via the communication protocol.

4.4.4 System Architecture

The simplified architecture, of the new system is dominated by three independent personal computers. Each component, namely the TCU, the RCU, and the RCA run on one of their own. The system and its corresponding inputs and outputs is depicted in Fig. 4.5.

Next, we discuss the system requirements of the prospective RCA tool.

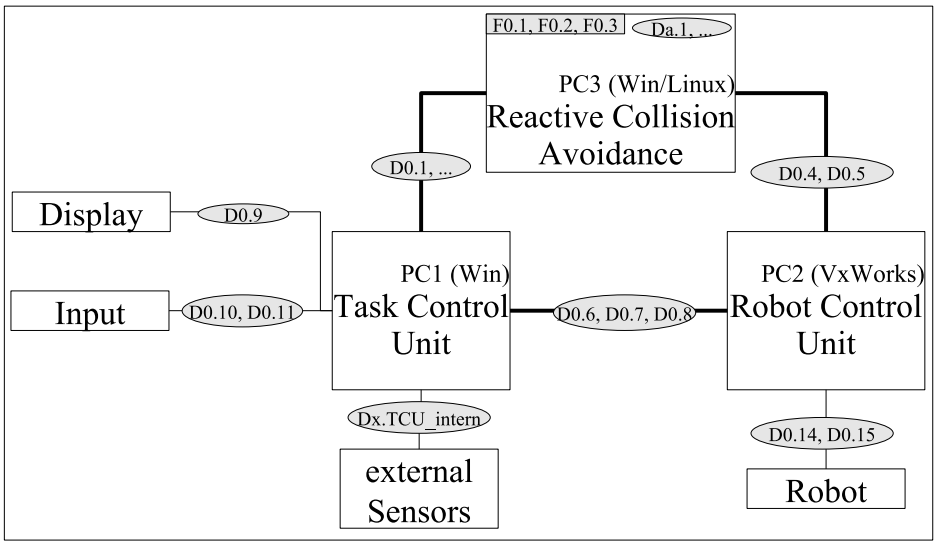


Figure 4.5: Architecture Diagram of the simplified system.

4.5 System Requirements

With the structured analysis we gained significant insight into the system. Its primary terminators, new functionalities, and its general interfaces were identified. With the help of the clarifying system view, the user requirements are reflected. This could be realized by taking into account also the information given by the state-of-the-art in Chapter 2 and the algorithmic foundations in Chapter 3. Therefore, the RCA complies with the real-time constraints, the environment representation, the desired motion behavior, independence of the tool, as well as the necessary interfaces.

These requirements are described in the following.

Processing in Real-Time All calculations performed by the RCA need to fulfill a minimum requirement of real-time behavior and calculation intervals. The quality of data is of course also directly coupled with the control rate. In order to achieve sufficient performance in the real-world setup, force and velocity control rates need to be at leas 50 Hz.

Dynamic and Complex 6D Objects The environment description shall provide AOs for the robot representation as well as OOs for multiple environmental objects after they have been detected. We aim at implementing AOs for the representation of

- the robot end-effectors,
- the robot forearm and its upper arm.

For the obstacle representation we decided to generate the following objects.

- balls,
- ellipsoids,
- tube or bar elements,
- planes,
- and prismatic polygonial elements.

If available the properties of the generated objects should also include inertial and the corresponding force response behavior.

6D Task Motion The AO representing the end-effector should perform a stable and Task orientated motion.

Standalone Application The RCA is supposed to work as a standalone application. In order to attach the RCA to the TCU and RCU we use the proprietary communication protocol ArdNet, an DLR network communication tool that is fully integrable in Matlab/Simulink and works across distributed computer systems. Furthermore, the implementation should be easily extendable without needing to take the corresponding changes into account on RCU or TCU side.

Interface Design The interface design shall fulfill to complete the requirements of **D0.1** up to **D0.5**, as defined in Sec. 4.4.3. Therefore, an interface for updating the data of AO and OO states, as well as command interfaces for object generation is to be designed. The output interface to the RCU needs to be able to provide desired force-torque, velocity, or posture values to control avoiding objects. For the TCU the visualization of virtual force data is of concern.

5 Software Design

According to the structural analysis given in Sec. 4.5 and the corresponding elaborated requirements we describe the developed models and the final implementation in this chapter. In general, the implementation is based on the main functionalities outlined in Sec. 4.4.3:

1. the iteration and prediction of objects,
2. the object storage design,
3. generating, embedding, and updating of objects,
4. and the calculation of environmental forces.

The previously defined functionalities are split up into smaller functions. In addition, we describe the implemented functions that provide the infrastructure to run together with the RCU and TCU.

5.1 Overview

In this thesis models with increasing complexity are simulated and implemented. For testing and verification purposes we implemented point mass models according to (3.1) as avoiding objects (AOs) in 2D environments (line elements) and 3D environments (surface elements) for analyzing the particular algorithms.

For more complex implementations we used geometric hull robot models to associate the robot representation with a closed surface, i.e. a volumetric structure. Finally, 6D task environments with the according objects (including 6D dynamic behavior) were developed that can described as follows.

$$\begin{aligned} &\ddot{\mathbf{x}} = \mathbf{M}_x^{-1}\mathbf{F}_d && \mathbf{M}_x \in \mathbb{R}^{6\times 6} \\ &\mathbf{F}_d = \left[\mathbf{f}^T\ \mathbf{m}^T\right]^T = [f_1\ f_2\ f_3\ m_1\ m_2\ m_3]^T && \mathbf{F}_d \in \mathbb{R}^6 \\ & && \ddot{\mathbf{x}} \in \mathbb{R}^6 \end{aligned} \tag{5.1}$$

In the following we introduce the necessary environmental representations for the according realizations.

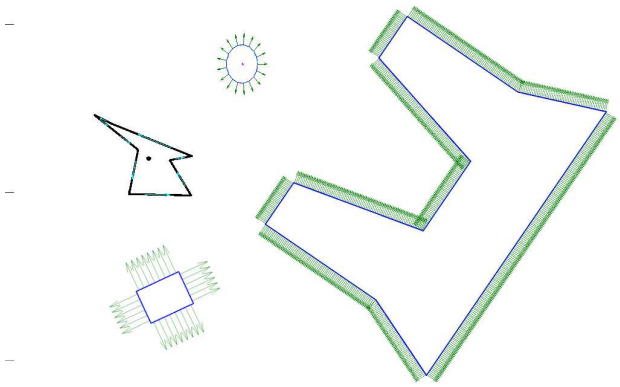


Figure 5.1: Examples of implemented 2D objects.

5.1.1 2D Representation

In 2D space line elements create obstacle objects (OOs) j that consists of

- vertices $\mathbf{p}_{x,j,i}$, which average value $\sum_i \mathbf{p}_{x,j,i}$ results in a center point $\mathbf{x}_{c,j}$,
- an adjacency matrix $\mathbf{CON}_j$ that is used to create the line elements $l_{j,i}$ with the associated normals $\mathbf{n}_{j,i}$ and the normal positions $\mathbf{x}_{j,n_i}$

The adjacency matrix is constructed such that a closed hull of the object is created and every line element length is $\approx \Delta l$, a design parameter that is defined by the user. In 2D following object types were implemented.

1. circular object,
2. cubic object,
3. random polygon object,
4. and more complex objects as U-trap, V-trap and a saw-profile

Fig. 5.1 shows examples of the aforementioned geometric types. Circular and random line polygonal objects are used in Sec. 6.1 to generate random testbed scenarios.

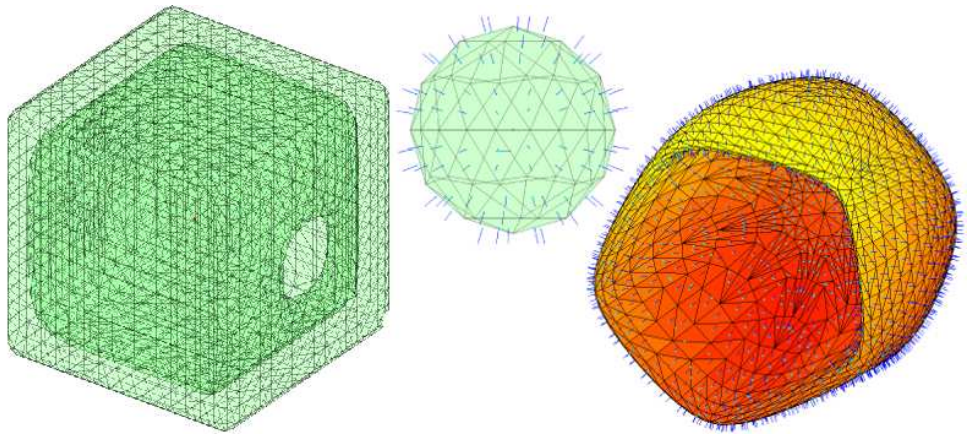


Figure 5.2: Collection of 3D objects.

5.1.2 3D Representation

For 3D environments static objects are created by point and surface elements. The corresponding data structure consists of following parts.

- Vertices $\mathbf{p}_{x,j,i}$ and adjacent matrices $\mathbf{F}_j$ containing surface representation. These are used for visualizing surface elements.
- Furthermore, each surface element is associated with its normal $\mathbf{n}_{j,i}$ and normal position $\mathbf{x}_{j,n_i}$ with average value resulting in a center point $\mathbf{x}_{c,j}$.
- Alternatively, a spherical as well as voxel space description[1] of objects is available.

5.1.3 6D Representation

Since up to now AOs were treated as simple point mass objects, we need a consistent description for the 6D case. This is due to the fact that the AO representation gets significantly more complex when assigning complex geometric objects with inertial properties. In order to achieve a modular system structure for the integration into the full robot control architecture we chose for both, OOs and AOs more or less identical data structures:

[1] In order to fully exploit the advantages of a voxel space representation a C implementation will be carried out in the future, since MATLAB/Simulink is of course not designed for this kind of data structures.

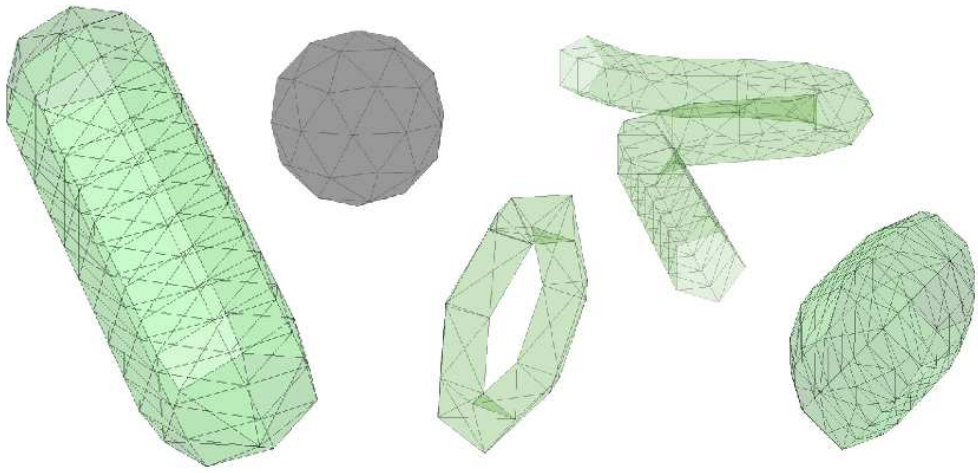


Figure 5.3: Collection of 6D objects.

For 6D environments point and surface elements create a local object (here shown for an obstacle j):

- Vertices $\mathbf{p}_{x,j,i}$ and surface adjacency matrices $\mathbf{F}_j$ containing surface representations are used for visualization of surface elements.
- Furthermore, each surface element is associated with its normal $\mathbf{n}_{j,i}$ and normal position $\mathbf{x}_{j,n_i}$ with average value resulting in a center point $\mathbf{x}_{c,j}$.
- Alternatively, a spherical as well as voxel space description of objects is available.
- Inertial parameters of object can be assigned.

Further data expressed in the world frame $S_W \in SE(3)$ that is required for force calculation and simulation is also stored and updated:

- Surface position ${}^W\mathbf{x}_{j,n_i}$, normals ${}^W\mathbf{n}_{j,i}$, velocity ${}^W\dot{\mathbf{x}}_O, j$, position ${}^W\mathbf{x}_{O,j}$, and forces ${}^W\mathbf{f}_{O,j}$ acting on the object.
- Transformation matrix ${}^W\mathbf{T}_{O,j}$, and object information as e.g. object type, force mode, data of posture and its velocity.

For the 6D case we implemented following obstacles, 3 AO types:

- End-effector (ring torus),

- Forearm, and
- Upper arm

In addition, n freely assignable OO types are available:

- Sphere,
- Ellipsoid,
- Wire elements,
- Bar elements,
- Plane and prismatic polygonic elements are planned, and
- Complex objects (trap and box).

5.2 Implementation

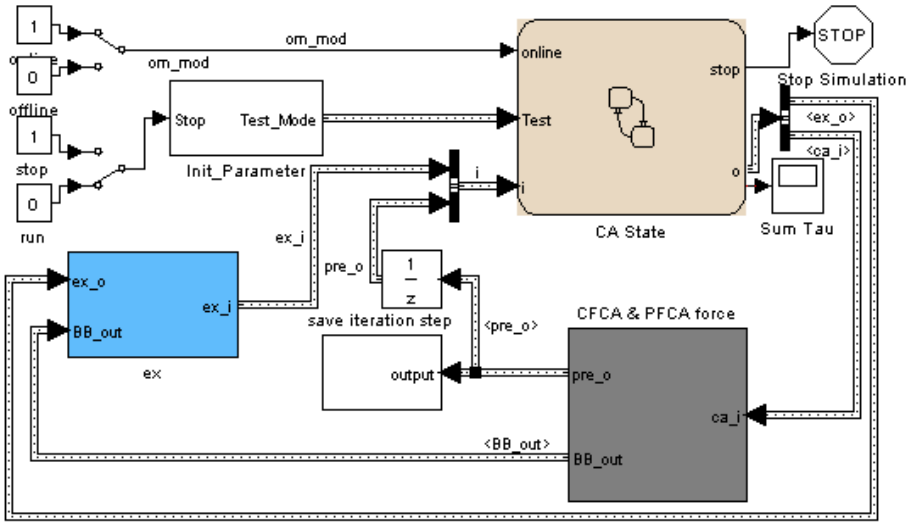


Figure 5.4: Overview of the Simulink model.

We describe now, the Simulink model for the implementation for 6D Simulation and the final experiments. Furthermore, we introduce the chosen

data structure, the pointer system for objects, and the most important sub-functionalities. The Simulink model is structured into three major blocks, see Fig. 5.4. The planning level is represented by the State Flow chart, the calculation of environment and forces by a block with embedded Matlab functions, and the communication to RCU and TCU is performed by the communication block that uses ArdNet[2] connections for communication. With this structure the State Flow chart and the embedded functions create a control loop. The communication exchanges data to allow updates of the virtual environment (RCU & TCU) and transmission of forces to the RCU for robot control. The State Flow chart consists of two parallel charts for communi-

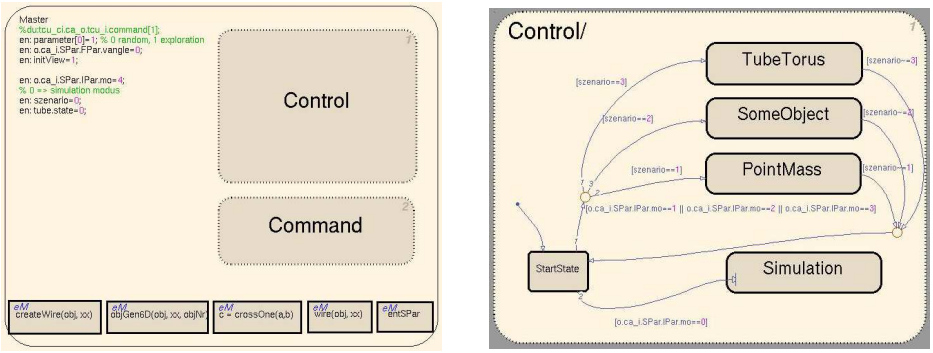


Figure 5.5: Overview of the State Flow charts.

cation and control, see Fig. 5.5. The communication chart is organized into two parallel charts that organize communication on command level for both, RCU and TCU. The control chart is used to implement respective control finite state machines for different scenarios. The embedded function block is subdivided into three sub-functions for iteration, force calculation, and prediction, see Fig 5.6. The iteration function is used for rotation and translation of objects into the current state in the virtual environment. The force calculation function uses different algorithms and combination strategies to generate forces. The third embedded function, the calculation function is used for simulation without TCU and RCU. This enables the prediction of desired velocity and position values without the need of running the full system. This function is also used to organize the output data for the TCU, so that the visualization of resulting forces can be done by already existing

[2] ArdNet is the standard network communication tool at DLR. It uses shared memories with separate input and output network connections for refreshing the allocated shared memory.

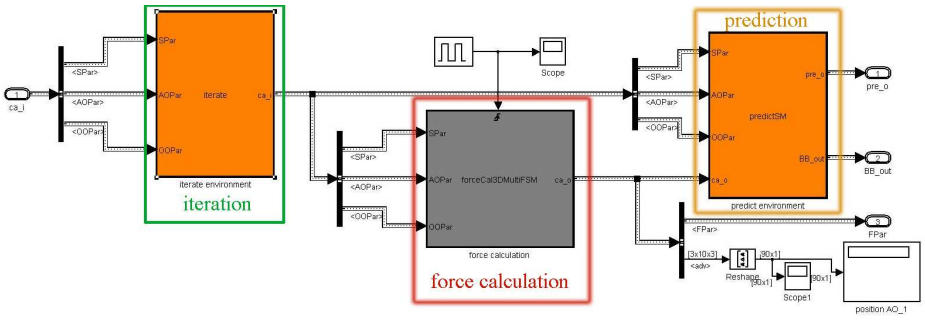


Figure 5.6: Overview of the Simulink Embedded Functions.

tools.

Comparison with requirements The presented structure provides the required functionality described in Sec. 4.4. The interrelations in the current model version are given in Fig. 5.7. The State Flow chart holds the functions

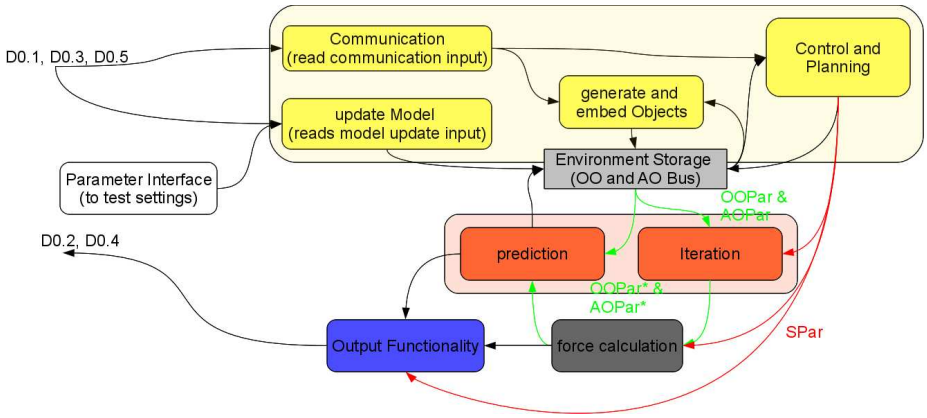


Figure 5.7: Overview of implementation with detailed functionality.

for communication on command level, update environment, generation of objects, and for high level control and planning. The chart fulfills the function **F0.2** and additional functionalities. The function **F0.3** is realized by the iteration, as well by the prediction function. The central entity "force calculation function" **F0.1** is realized as planned, whereby the high-level control and strategies are implemented in the State Flow chart. The output functionalities (blue) are at the moment still included in the prediction function. The entire static storage of the model as e.g. the initial object data and

the algorithmic parameters are hold by the State Flow chart as its output port. This port consists of three data structures, which are described in the following.

5.2.1 Data Structure

The chosen data structure of the current implementation is separated into structures for simulation or administration parameters *SPar*, OO data *OOPar* and AO data *AOPar*. When designing this data structures it was taken care of creating no variables with more than 3 dimensions in order to avoid the Mathworks compiler limitations[3]. The entire data structure is defined as non-virtual Simulink buses. These buses are also used in the embedded functions for sake of clarity.

We first describe the hold object data. The data structures of *AOPar* and *OOPar* are very similar. Therefore, only *AOPar* is described and the particular differences to *OOPar* are pointed out. Both variables consist of 12 fragmentations. These are described in the data dictionary of *AOPar* and *OOPar*, see Tab. 5.1.

Data Name	**Description**	**Dimension**	
		AOPar	OOPar
dv	- desired values of object (x, phi, fa, fd, fv, Ma, Md, Mv, v, omega)	$3 \times 10 \times n_{\mathrm{m}}$[4]	$3 \times 10 \times j_{\mathrm{m}}$[5]
J	- object transformation matrix	$4 \times 4 \times n_{\mathrm{m}}$	$4 \times 4 \times j_{\mathrm{m}}$
M_x	- mass matrix M_x of object	$6 \times 6 \times n_{\mathrm{m}}$	$6 \times 6 \times j_{\mathrm{m}}$
M	- inertia of object	$3 \times 3 \times n_{\mathrm{m}}$	$3 \times 3 \times j_{\mathrm{m}}$
N*	- initial object description	1	1
n_c	- number of object surfaces	$1 \times n_{\mathrm{m}}$	$1 \times j_{\mathrm{m}}$
N_xA*	- updated surface description	1	1
n_cA	- number of currents influencing object surfaces	$-*- \times n_{\mathrm{m}}$	$-*- \times j_{\mathrm{m}}$
continues next page ...			

[3]The acceleration of arrays with more than 3 dimensions is not supported for m-files. We assumed the same to be true for embedded functions to prevent problems in the implementation phase.

[4]n_{m} is the maximum number of AOs

[5]j_{m} is the maximum number of OOs

Data Name	Description	Dimension	
		AOPar	OOPar
asp	- actual surface pointer on influenced or influencing object surfaces	$-*- \times k_{\mathrm{m}}$[6]	$-*- \times i_{\mathrm{m}}$[7]
mo	- object mode (static, movement, rotation → includes movement)	$1 \times n_{\mathrm{m}}$	$1 \times j_{\mathrm{m}}$
kind	- object type	$1 \times n_{\mathrm{m}}$	$1 \times j_{\mathrm{m}}$
vox*	- not used - voxel space representation for objects	1	1

Table 5.1: Data dictionary of structure variable AOPar and OOPar.

Three of these fragments include sub-fragments that are marked by *. Fragments that are not fully defined are marked as $-*-$ for undefined dimensions, or described as - not used - if the development of the RCA has led to not using this variable anymore. The overall structure holds more than the data necessary to calculate exclusive forces, velocities, or position data in order to be able to easily extend it if needed. The sub-fragment *vox* of *AOPar* is described in detail as it is currently not in use. This voxel-space representation uses a linked list (see Sec. 5.2.4) to hold multiple surfaces at one voxel grid. The sub-fragments *N* and *N_xA* are described in detail in Tab. 5.2 and Tab. 5.3.

Data Name	Description	Dimension	
		AOPar	OOPar
c	- fix configuration points: center of mass or transformation base	3×3	3×3
n	- initial surface normals of an object	$4 \times k_{\mathrm{m}} \times n_{\mathrm{m}}$	$4 \times i_{\mathrm{m}} \times j_{\mathrm{m}}$
dx	- surface position vectors from transformation base or center of mass	$3 \times k_{\mathrm{m}} \times n_{\mathrm{m}}$	$3 \times i_{\mathrm{m}} \times j_{\mathrm{m}}$
face	- surface-vertex adjacency polygons	$k_{\mathrm{m}} \times 3 \times n_{\mathrm{m}}$	$i_{\mathrm{m}} \times 3 \times j_{\mathrm{m}}$
dP	- vertex position vectors from transformation base or center of mass	$3 \times k_{\mathrm{m}} \times n_{\mathrm{m}}$	$3 \times i_{\mathrm{m}} \times j_{\mathrm{m}}$
r	- physical range of object	$1 \times n_{\mathrm{m}}$	$1 \times j_{\mathrm{m}}$

Table 5.2: Data dictionary of structure variable N.

[6] k_{m} is the maximum number of surfaces of an AO
[7] i_{m} is the maximum number of surfaces of an OO

The data structure of N describes is used as basis for redefinition of N_xA by the transformation matrix T_j. The detailed description of N_xA is given by Tab. 5.3.

Data Name	Description	Dimension	
		AOPar	OOPar
fc	- force vectors of configuration frame	$6 \times 3 \times n_{\mathrm{m}}$	$6 \times 3 \times j_{\mathrm{m}}$
xc	- position vectors of configuration frame	$3 \times 3 \times n_{\mathrm{m}}$	$3 \times 3 \times j_{\mathrm{m}}$
vc	- velocity vectors of configuration frame	$3 \times 3 \times n_{\mathrm{m}}$	$3 \times 3 \times j_{\mathrm{m}}$
moc	- number of objects	$-*- \times n_{\mathrm{m}}$	$-*- \times j_{\mathrm{m}}$
f	- surface force vectors	$6 \times k_{\mathrm{m}} \times n_{\mathrm{m}}$	$6 \times \times i_{\mathrm{m}} j_{\mathrm{m}}$
n	- surface normals of an object	$4 \times k_{\mathrm{m}} \times n_{\mathrm{m}}$	$4 \times i_{\mathrm{m}} \times j_{\mathrm{m}}$
x	- surface position vectors	$3 \times k_{\mathrm{m}} \times n_{\mathrm{m}}$	$3 \times i_{\mathrm{m}} \times j_{\mathrm{m}}$
v	- not used -surface velocity vectors	$3 \times k_{\mathrm{m}} \times n_{\mathrm{m}}$	$3 \times i_{\mathrm{m}} \times j_{\mathrm{m}}$
vox*	- not used - voxel coordinates	1	1
mo	- not used - mode of surface forces	$-*- \times n_{\mathrm{m}}$	$-*- \times j_{\mathrm{m}}$

Table 5.3: Data dictionary of structure variable N_xA.

In addition to the position and normal configuration expressed in the world frame, N_xA is able to store forces (**fc**,**f**), velocities (**vc**,**v**), a separated voxel space representation, as well as information of force modes of every surface. In this structure a significant amount of memory is allocated, which is, however, relatively static and less frequently accessed. Next, we describe the fragment *SPar*. *SPar* consists of 18 fragmentations. Four of these fragments also include sub-fragments as shown in the data dictionary of *SPar* in Tab. 5.4.

Data Name	Description	Dimension
da	- surface segment size	1
IPar*	- iteration parameters	1
FPar*	- force parameters	1
objnum	- number of objects	1
objp	- pointer between outer and inner representation	$2 \times (j_{\mathrm{m}} + n_{\mathrm{m}})$
onum	- number of OOs	$1 \times n_{\mathrm{m}}$
continues next page ...		

Data Name	Description	Dimension
anum	- number of AOs	1
io_c	- counter of influenced OOs	n_m
ia_c	- counter of influenced AOs	j_m
op	- pointer on actual inserted OOs	$n_\mathrm{m} \times j_\mathrm{m}$
ap	- pointer on actual inserted AOs	$(j_\mathrm{m} + 1) \times n_\mathrm{m}$
iop	- pointer on actual influencing OOs (of an AO)	$n_\mathrm{m} \times j_\mathrm{m}$
iap	- pointer on actual influencing AOs (of an OO)	$j_\mathrm{m} \times n_\mathrm{m}$
areaTrigger	- not used -	1
mdg	- not used -	1
situation	- stores start goal configuration	3×2
Limits*	- stores dimensions of variables and defined sizes, e.g. voxel space	1
Test*	input variable for test parameters	1

Table 5.4: Data dictionary of structure variable SPar.

The sub-fragment *IPar* stores iteration data as the simulation step size and limitation of motion speed. *FPar* stores force and algorithmic parameters. *Limit* holds the variable and environmental dimensions and *Test* the input data from the parameter input interface, which was sketched in Fig. 5.4 and 5.7. If the parameter input interface is active, parameters of *IPar* and *FPar* can be changed online. The remaining *SPar* consists primarily of counters and pointers that construct the infrastructure, addressing the external representation with internal storage and usage. For further understanding the implemented pointer system is explained in the following.

5.2.2 Pointer System

For the communication with a client as e.g. the TCU it is important to hold all objects by an invariant communication interface. If the client intends to generate an object, the client only be necessary to send the respective only command. In turn, it should receives the object address and/or ID. By using this address a client may easily change or delete an object without the need to knowing the internal address. The address is hereby provided by an object counter **obC**, which increases by one for every newly generated object. Several variables hold the total numbers of objects (objnum), the

number of every kind of objects, obstacle or AOs, and the links on storage, see Fig. 5.8. Furthermore, they also hold some links to other pointer tables

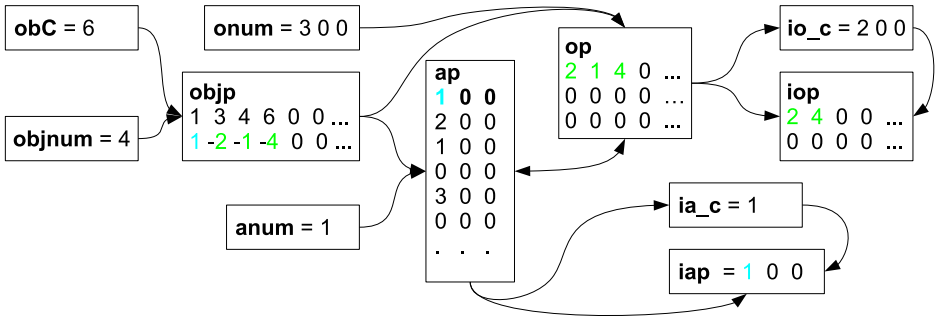


Figure 5.8: The directed graph of the internal pointer system.

for reorganizing the pointer system if objects need to be integrated. Some pointers as **io_c**, **iop**, **ia_c**, and **iap** hold the current influencing status of the objects (see e.g. OOs 2 and 4, which influence AO 1, Fig. 5.8). Using the **op** pointer, a limitation of the influencing objects for an AO can be assigned. The detailed descriptions of the pointer system elements is as follows.

- **obC** → object counter and address provider of all ever generated objects
- **objnum** → number of objects in the entire environment 1x1
- **objp** → a pointer that stores the external object address provided by **obC** in the first row and the internal links in the second row
- **anum** → number of existing AOs 1x1
- **ap** → AO pointer. In the first row the link for the AOs in the storage is given and in the next rows the influencing objects are linked to the **op**
- **onum** → number of OOs influencing an AO (e.g. 1x3)
- **op** → OO pointer. In every row the links in storage of the possible influencing objects are given.
- **io_c** → real influencing OOs counter, numbers out the obstacle that currently influences the AO

- **iop** → the storage links of influencing objects
- **ia_c** → influenced AO counter (not used yet)
- **iap** → storage link to the influenced AOs (not used yet)

The already given example in Fig. 5.8 can be considered for further description. With the pointer list **objp** one avoiding and 3 OOs are hold. The firstly generated object was the AO stored at position one of the AOs storage. The second and fifth object are already deleted and are now stored OOs in storage position 1, 2 and 4. The addresses 4, 3 and 6 are then returned to the TCU.

After describing the data structure and its object pointer system, the most important functionalities of the implementation are described in the following.

5.2.3 Communication and Command Control Design

The communication interface of the RCA for the second Co-Worker scenario (see Sec. 4.1 for general introduction) has to provide connections to 3 RCUs and one TCU. These communication lines and the chosen communication structure are described in this section. A separated communication line with fixed ArdNet connections is provided for every robot and the TCU. The three RCU communication lines are identical by means of their data structure. Therefore, only two structures are described. Both structures of RCU and TCU are subdivided into a permanent data-flow of robot or environmental data and a command line. The communication lines are illustrated in Tab. 5.5.

In the chosen design, the data command line of the RCU is much smaller than for the TCU. Also the achieved data volume of the robot arm is less than for the environment that is provided by the TCU. For every RCU three dynamical objects with posture and dynamical variables as velocity and angle velocity are estimated. For the TCU the basic version is estimated with ten dynamical objects, having the same data as the objects from the RCU. Furthermore, it is also possible that the communication lines are used for differing data. Instead of the dynamic state of the three AOs, the RCU could provide the desired and real posture, as well as force data of one AO for the RCA. This possibility is used in Sec. 6.3.4. For describing the communication protocol the following example “communication by command” is used. The basis of a command consists of four elements: *cmod* the command mode, *com*

Connection	Description	Size (byte)	Structure
from RCA to TCU	- number of forces	4	nr
	- 30 forces or torques with position	720	$[\mathbf{f}_{v,1}; \mathbf{x}_1; \mathbf{f}_{v,2}; \mathbf{x}_2 ...]$
	- RCU2TCU command	128	com=32 double
		=852	
from TCU to RCA	- position, velocity, orientation of 10 objects	880	[T + d$\mathbf{x}$]*10
	- TCU2RCA command (object initializing, set new goal)	400	com=100 double
		=1280	
from RCA to RCU	- forces & torques for 3 AOs	72	$[\mathbf{F}_{\text{env}}]$*3
	- velocity & angle velocity for 3 AOs	72	$[\mathbf{v}, \omega]$*3
	- position & angle for 3 AOs	72	$[\mathbf{x}, \phi]$*3
	- RCA2RCU command	128	com=32 double
		=344	
from RCU to RCA	- position, velocity, angle velocity & orientation for 3 AO	264	[T + d$\mathbf{x}$]*3
	- RCU2RCA command (object initializing, set new goal)	128	com=32 double
		=392	

Table 5.5: Data structure of communication.

the kind of command, the third open variable field, and *msgnr* the message number. *cmod* is in the following set equal 2, meaning that an object will be generated. *kind* and *com* are determine the object type and whether it is an OO (kind < 0) or an AO (kind > 0). In the example an ellipsoid is generated. The overall definition and the command structure is shown below.

```
cmod=2
Ellipsoid or Ball kind=3 or -3 => com=2
Description:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% description of an ellipsoid Ball object %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   R11     R12     R13     x       kind    rho
%   R21     R22     R23     y       s_x     mo
%   R31     R32     R33     z       s_y     -
```

```
%   -       -       -       -       s_z     -
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   R       ->  initial rotation
%   x y z   ->  initial position
%   mo      ->  object mode 0->O, 1->A&O, 2->A&O
%               (0 - static, 1 - moving, 2 - rotating)
%   kind    ->  kind of object in this case 3 or -3
%   rho     ->  density of volume which result in a mass
%   v       ->  velocity of object
%   s       ->  size of the object in x y z before rotation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Command:
cmod    R11     R12     R13     x       kind    rho     -
com     R21     R22     R23     y       s_x     mo      -
-       R31     R32     R33     z       s_y     -       -
msgnr   -       -       -       -       s_z     -       -
```

The object is defined by its initial posture, its density *rho* that allows to calculate its inertia, and the size in x, y and z direction. If only one size axis is defined a ball with certain radius is assumed and for more axes an ellipsoids is created. Another important part of every command is the message number *msgnr*. It is used to achieve the handshake with the client, which is necessary for fast and safe communication via the asynchronous ArdNet protocol. Furthermore, several commands are especially implemented from TCU to RCA. Exploiting the full range of manipulation possibility is left for the future. Next, we describe the several utility functions that were designed.

5.2.4 Utility Functions

In this section useful functionalities developed for the RCA are described. Standard functions as e.g. the reset functionality are not considered any further. Here, we focus on functions that are needed to run the RCA as e.g. for object generation, embedding, reading out of files, and other illustrations. First, we present the object generation functionality in the following.

Generation and Embedding of Objects

In general, the generation and embedding of objects is combined into one function. This can be separated into the sub-functions of embedding the

new object in the pointer system, generating the local representation, embedding the object information in the existing structure, and activating the object for calculation and simulation. Embedding into the pointer system can be understood best by consulting Sec. 5.2.2 for the pointer system again. The procedure to create OOs or AOs is very similar and both are therefore described together. For object generation the vertices and adjacency matrix of the object are read or generated first. The surfaces and corresponding normals are then created. Furthermore, parallel dynamic data is calculated or read. This basic representation is then transformed into the initial frame and stored in the according structure. After embedding the local frame, the object is transformed into the possibly given start state in the global frame, embedded and initialized e.g. with the mode and object type. In general, we generate objects from basic geometric information or read externally created objects. Objects that are too large need to be partitioned into multiple parts. This is e.g. realized for the wire elements used in Sec. 6.3.3. The function that can be used to load objects also from files is described in the following.

Read Function for Inventor-Files

While implementing multiple scenarios for simulation, the import functionality for inventor files was also created. For example the objects simulated in Sec. 6.2.1 and 6.3.1 were imported by this function. The read function is especially of interest for externally provided OO or AO inventor files.

Object Delete Function

We described how objects are generated, read, and embedded in the RCA. It is also necessary to delete these objects if they are not required anymore. Therefore, a function that is also externally accessible was implemented for this purpose (also for deleting multiple objects at the same time). First, the objects are localized by their external address as already explained in Sec. 5.2.2. With a list of addresses, several objects and by a special command mode all OOs or AOs can be deleted. For deleting an object a sub-function deletes first the representation in the pointer system and reorganizes it. Then, the entire object information in local and global frame is reseted.

After describing the generation, import, and erasing of objects, the update and output functionality is presented in the following.

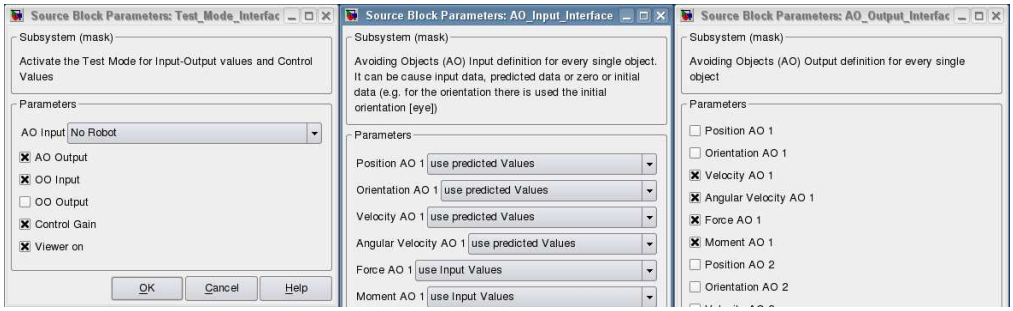


Figure 5.9: Overview of the input output interface for AOs.

Update and Output Functionality

The update and output functionality is integrated in multiple parts of the RCA to provide freely assignable input and output behavior. Therefore, the parameter interface (see Fig. 5.4 "Init_Parameter" block) is also used as an interface for this functionality to provide online input and output changes by selection and Drag & Drop, see Fig. 5.9.

As the input for every AO (in the middle) we may choose the position, orientation, translational velocity, angle velocity and external force-torque data. These are then updated with input, predicted, or static (for dynamic zero) data. Analogously, the output data for the RCU can be chosen (right). In order to do so, the interface (left) for defining input and output of AO is set active, elsewise the internal set input and output are used. The interfaces for OO and the assigned robot are arranged in the same manner. The selectable robots are Eric, Jimi, and Eddie.

Voxel Space Illustration

The voxel[8] space illustration was implemented to provide a fast interface for finding links to surfaces within the range of influence for an avoiding particle. This functionality is still running with the actual object representation and is of interest for further development of a deterministic process of force calculation. This will allow the calculation of the theoretical runtime $O(.)$. For this, multiple surfaces can e.g. be mapped to a discrete voxel grid in order to achieve a deterministic representation of the environment. For every voxel this representation can e.g. be an average surface normal and the sum of

[8]The volumetric picture element is a volume element that represents a value on a regular grid in three dimensional space.

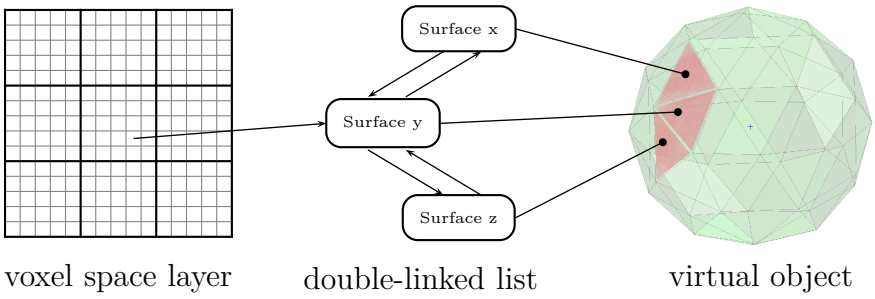


Figure 5.10: Sketch of the implemented voxel space representation.

areas. A dynamic update of the voxel space may be achieved in parallel to the force calculation.

The actual voxel space illustration maps the surface address (pointer) into the voxel space. For every voxel only one address is illustrated. For every surface an element of a doubly-linked list is allocated according to the actual voxel. Therefore, the list element (surface) has a status as being either first element, element in the middle, or last element. In addition, the pointer to the a previous and next element are stored, see Fig.5.10. Using this approach the memory allocation requirement is deterministic and all surfaces associated to a voxel can be found.

6 Analysis: Simulations and Experiments

In this chapter the simulational and experimental results we obtained are presented and analyzed. They are arranged by means of rising complexity. In the 2D simulations we compare the capability of the previously selected algorithms with each other. In 3D simulations and the according experiments the rising complexity of implementation and motion generation strategies is evaluated. Finally, 6D simulations and rather complex experiments on the LWR-III are presented, which confirm the feasibility and usefulness of the algorithmic implementations.

First, the possible extension of robot control by an virtual environment observer should be described to give an overview of the available control interfaces that may be used for disturbance input signals as e.g. virtual forces.

Control Values Impedance control enables us to use force, velocity, or position reference values to control the robot. Due to the physically interpretable behavior the force interface is advantageous for manipulating the reference motion.

For forces control schemes real external forces can be utilized to react to unforeseen collision forces, or desired virtual forces can be generated by a virtual environment observer and directly fed as desired force values. For the case that an external or desired forces $\mathbf{F}_d$ directly influences the end-effector this force is directly added to the desired control force.

$$\mathbf{F} = \mathbf{F}_d + \mathbf{F}_{control} \ . \tag{6.1}$$

For such a virtual environment force the robot is therefore directly affected by an external dynamics as e.g.

$$\mathbf{F}_d = \mathbf{F}_a + \mathbf{F}_D + \mathbf{F}_v \quad \text{with} \quad \mathbf{F}_v = \sum_j \mathbf{F}_{ob,j}, \tag{6.2}$$

where $\mathbf{F}_a$ is an attractor force, $\mathbf{F}_D$ is a damping element, and $\mathbf{F}_v$ is the resulting virtual force of the obstacles contained in the virtual environment. Figure 6.1 depicts general force input variants that may be used. As already

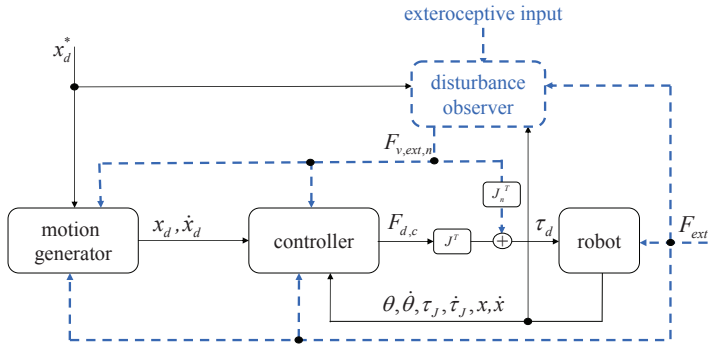


Figure 6.1: Control principles for force input.

mentioned we may use the implemented torque and position controller. External forces are also estimated with a nonlinear disturbance observer and can be used by position and velocity controllers or for creating an outer-loop that implements a certain dynamic behavior (denoted by attractor). In this thesis we utilize a particular attractor type that was recently developed [21]. Its main characteristics are that is absolutely ensures a desired absolute velocity profile (e.g. constant velocity) and alters its stiffness behavior depending on virtual or real disturbances as e.g. external forces, see Fig. 6.2. The input variables for the attractor are external as well as virtual forces

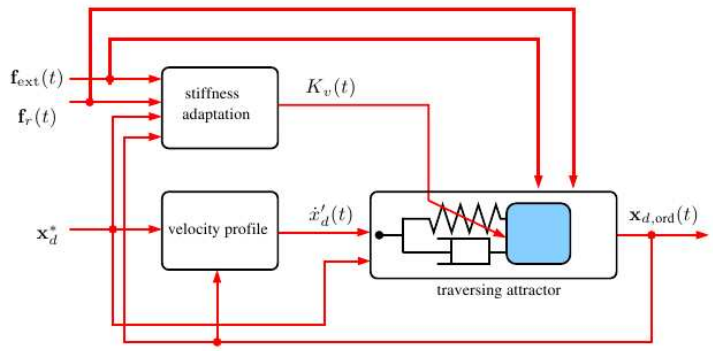


Figure 6.2: Attractor with velocity scaling and integrated dynamic.

and the final desired position $\mathbf{x}_d^*$. Its overall behavior is determined by the associated impedance and the desired velocity profile. In this thesis the virtual environment force input is either used to directly influence the attractor (third dynamics) or is directly added to the desired control force (second dynamics). If the virtual environment observer includes a complete dynamics

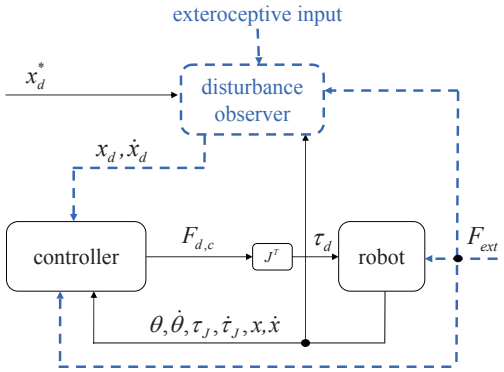


Figure 6.3: Control principle for position or velocity input.

model of the controlled robot (alternatively only for the end-effector) and provides desired position and velocity data these can be directly used as the reference trajectory for the low-level controller, see Fig. 6.3.

6.1 2D Simulation

In this section we review the simulation of the potential fields, circular fields, and optical flow methods and analyze them accordingly. This leads to a selection process for further evaluation.

6.1.1 Comparison HNPF and PF

Human navigation potential fields (HNPFs) (see Sec. 3.1.2) and potential fields (PFs) (see Sec. 3.1.1) are compared first, however, only briefly.

Both methods are based on the same gradient field and are not local minima free. Furthermore, they show almost the same results for U-objectcluster and dead-ends. Due the quite complicated parameterization of the HNPFs to minimize local minima, while still avoiding obstacles we conclude that the benefit of HNPF is only marginal compared to PFs and therefore rather analyze the classical potential field approach from now on.

Next, we compare the behavior of PFs CFs.

6.1.2 Comparison PF and CF

PFs and CFs are compared in a 2D randomly created environment, see Fig. 6.4. It consists of circular objects j assigned with line element nor-

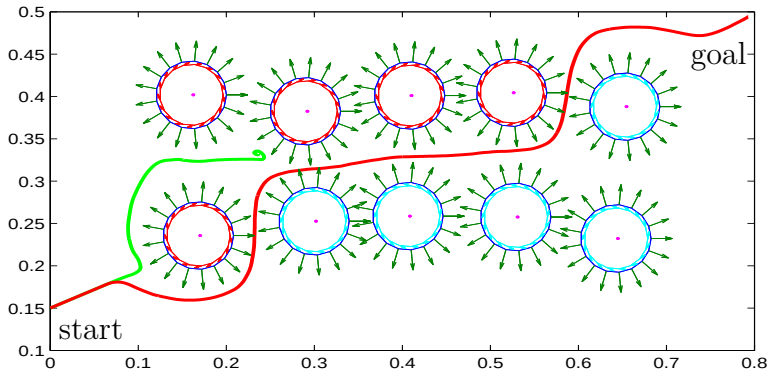


Figure 6.4: Comparison of Circular Fields and Potential Fields.

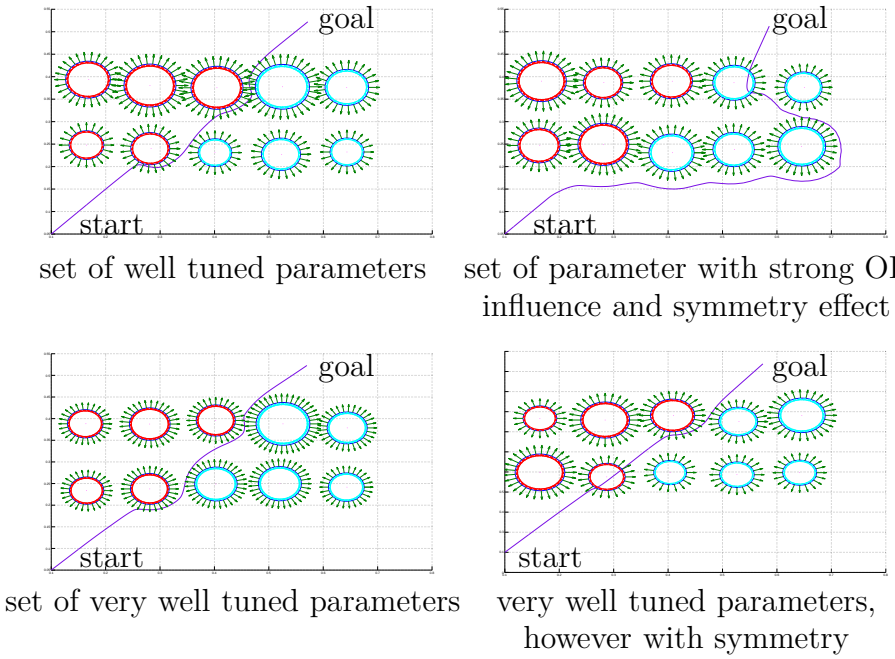
mals (dark green arrows). The start configuration is located in the lower left corner and the final goal in the upper right corner. The light green trajectory is generated via the potential fields and the red trajectory with the circular fields approach. The red marked obstacles having a right and the cyan marked obstacles having left turning CF rotation vectors $\mathbf{r}_j$. It becomes clear that PF based motion generation gets easily stuck in local minima and that the generated path has repulsive character. In contrast, the path obtained via CFs is very smooth and "graceful". The algorithms are equipped with a local range of 0.06 units and even with this limited range the CFs approach appears to be very robust. The shown CF algorithm uses the velocity angle adaptation introduced in Sec. 6.1.4 and a high gain I_K. Therefore, the point mass does not come closer than 0.05 to the obstacles except when moving through the parcour. Based on these results we only investigate the capabilities of CFs in further 2D simulation.

Next, CFs are compared with the OF approach.

6.1.3 Adapted Optical Flow

The angular optical flow (see Sec. 3.1.5) is next in simulation with two adaptations. The typical and already sketched adaptation (see Sec. 3.1.5) is achieved by rotation of the velocity vector or motion direction. The used rotation R of the AO velocity vector $\dot{\mathbf{x}}$ according to the angular OF magnitude is defined as

$$\dot{\mathbf{x}}_{\text{new}} = \mathbf{R}\left(k_1(\dot{\psi}_{\text{goal}} + \psi_{\text{goal}}) - k_2(\psi_{\text{goal}})\sum_j OF_j\left(\frac{\dot{\mathbf{x}}_{\text{rel,j}}}{\|\mathbf{x} - \mathbf{x}_{j,\text{obst}}\|}\right)\right)\dot{\mathbf{x}}, \quad (6.3)$$



set of well tuned parameters | set of parameter with strong OF influence and symmetry effect

set of very well tuned parameters | very well tuned parameters, however with symmetry

Figure 6.5: Optical Flow simulation examples.

where $\dot{\mathbf{x}}_{\text{new}}$ is the reorientated velocity vector, ψ_{goal} is the angle between motion and goal direction, $\dot{\mathbf{x}}_{\text{rel,j}}$ is the relative velocity between AO and OO, and $\mathbf{x}_{j,\text{obst}}$ is the position of the OO. k_1 and $k_2(\psi_{\text{goal}})$ are controller gains.

The second adaption by rotation of the attracting force $\mathbf{f}_a$ is defined as

$$\mathbf{f}_{a,\text{new}} = \mathbf{R}\left(-k_3(\psi_{\text{goal}})\sum_j OF_j\left(\frac{\dot{\mathbf{x}}_{\text{rel,j}}}{\|\mathbf{x}-\mathbf{x}_{j,\text{obst}}\|}\right)\right)\mathbf{f}_a, \tag{6.4}$$

where $\mathbf{f}_{a,\text{new}}$ is the reorientated attracting force and $k_3(\psi_{\text{goal}})$ is a gain factor. The force approach is chosen be able to interface virtual forces as e.g. generated by an attractor for free space or for running in combination with other algorithms that are based on virtual forces. The free space approach was tested.

In general the motion direction and force approach realize very similar behavior. Therefore, we use the force approach for further description. As already in the previous comparison we generate random scenarios with circular objects to verify the approaches, see Fig. 6.5.
In Fig. 6.5 all obstacles are marked with CF rotation directions $\mathbf{r}_j$ so that the corresponding motion with CFs can be visualized. The OF trajectory is

depicted in purple. The setups *left* and *lower right* depict simulation results with variable environment and settings of well tuned parameters. This was necessary to obtain results with similar quality as with CFs. However, in the special case *lower right* the same parameters lead to a collision due to the symmetric setup. *Top right* shows a result with strong influence of OF. This leads for the point mass to a tendency to move around the entire obstacle field. Furthermore, the symmetry failure is observed again.

In general, OF tends to become zero in an environment with many symmetries and the point mass moves therefore through the virtual obstacles. Furthermore, it is not trivial to find parameters for dynamic environments and more complex scenarios. One may assume that the symmetry problems decrease when using direct sensor input. Unfortunately, this cannot be verified at this point. However, the simulation results achieved with OF are of significantly lower quality compared to CFs. So to sum up, despite reasonable effort it was difficult to generate successful motion generation with OF based schemes and therefore, we exclude this algorithm as well from our further investigations.

6.1.4 Static CF Examples

In this subsection we discuss varying simulation results obtained with CFs to validate their capabilities for further use. Simple geometric obstacles in varying situations, as well as more complex objects were simulated.

Basic Simulation of CFs

In Sec. 3.1.4 it was already shown that CFs are strictly converging to the respective goal, since they are free of local minima. Furthermore, they are intrinsically collision free if the currents on the obstacle surfaces are defined correctly. The results for some further simulation environments are depicted in Fig. 6.6.

Top left: shows a simulation of a cubic object that is avoided by a point mass via CF obstacle forces (yellow arrows). The simulation also incorporates a damping and an attractive force (cyan arrows) (compare to the basic approach described in Sec. 3.1.4). Similar to the previous results a smooth and strictly collision avoiding trajectory is created. An important observation is that the point mass is captured by the CFs also after passing by and directly heading towards the final goal.

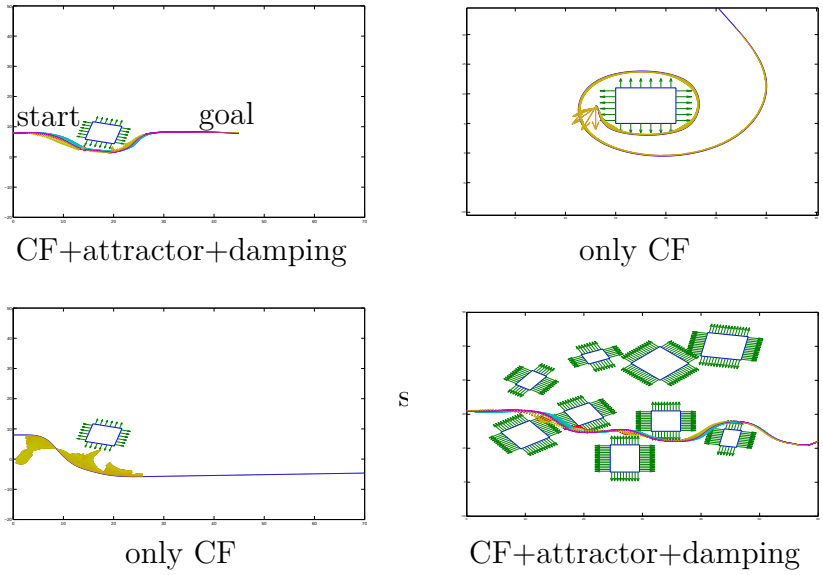


Figure 6.6: Basic simulation of CFs.

Down right: shows a similar simulation with multiple random positioned cuboids.

Lower left: shows the same situation as above without damping and attracting forces. The point is initialized with its start velocity in goal direction. Its final velocity direction is approximately the same as for the start point and the goal is of course not reached. However, the obstacle is avoided by a smooth trajectory. Please note that since no energy is dissipated the magnitude of the resulting velocity $\|\dot{\mathbf{x}}\| = \text{const}$.

Top right: shows a simulation without damping and attractive forces. However, the starting point of simulation is in the range of influence of the obstacle. Obviously, the CFs do not capture the mass point despite the absence of an attractor.

Simulation of Velocity Angle Adaptation The adaptation described in Sec. 3.3.2 is discussed now. The simulation results are depicted in Fig. 6.7. The blue trajectory is not influenced by the velocity rotation $\psi = 0$. For the black trajectory we choose $\psi = 60°$. Clearly, for ψ between 10° and 30° a smooth and intuitive trajectory is generated. The point mass derivates significantly earlier in front of the obstacles and is therefore also not captured that much after passing the obstacle.

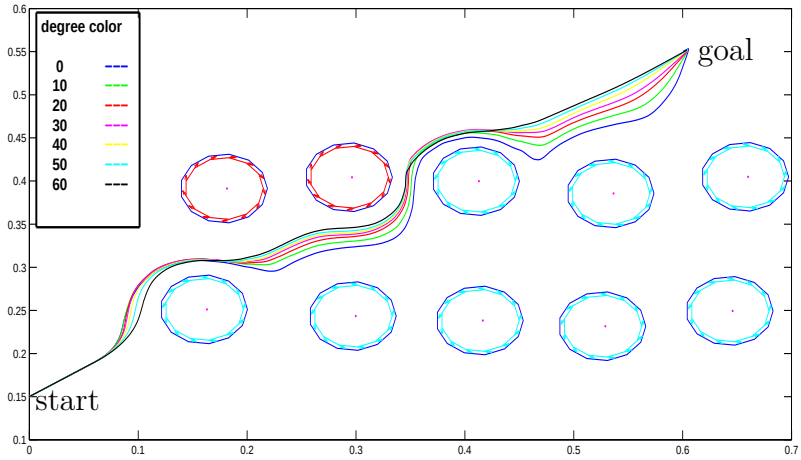


Figure 6.7: Deviation of a trajectory by using variable angles for the velocity angle adaptation.

Simulation Random 2D

For more variability automatic generated random testbeds were simulated to identify problems in environment representation or special situations that are leading to collisions. A particularly interesting example of these random testbed simulations is shown in Fig. 6.8. It depicts a typical narrow passage problem. The polygons are randomly generated and six sample steps from the full simulation were chosen to indicate the performance of the method. The virtual particle has a limited view range, which is indicated by the virtual forces calculated for the discretized surface elements. The resulting external force (red arrow) acting on the virtual particle and the attractor force (green) are shown. Furthermore, the associated CF rotation direction is indicated on the object border. This corresponds to an opposite direction of the current. The example clearly shows that even with this sharp-edged obstacles the algorithms is smoothly guiding the point mass to the goal.

Static complex 2D examples

For further capability analysis we simulated more complex objects as the ones shown in Fig. 6.9. The *upper left* image depicts a saw profile problem. The generated motion also visualizes the capture effect that is generated by CFs. For pointing out this effect more clearly only the sum of forces (red arrows) is illustrated for every simulation step. In order to pass difficult traps, very strong CFs or a modified attractor lead to a convenient behavior.

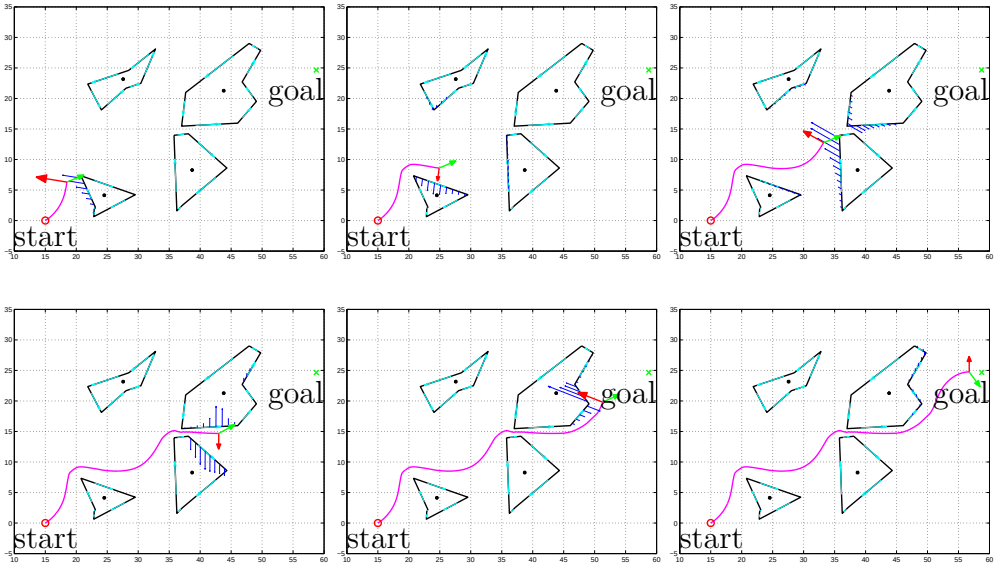


Figure 6.8: 2D example for the circular field method surpassing a narrow passage.

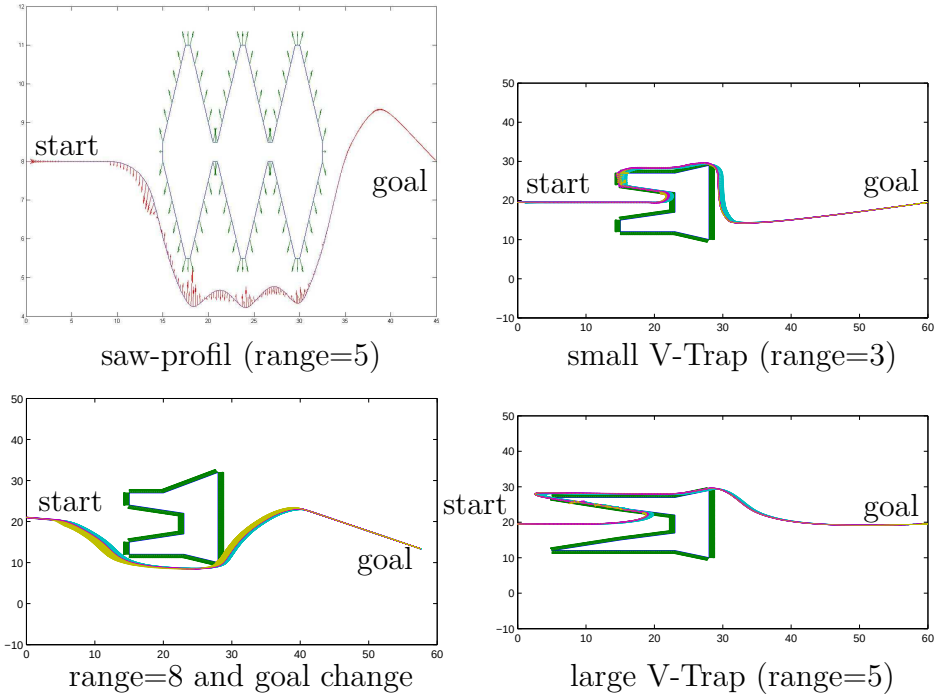


Figure 6.9: Avoidance of complex 2D obstacles with point mass.

The redefined attracting force incorporates the weight of normals $\mathbf{n}_i$ in range and the current velocity direction $\dot{\mathbf{x}}$.

$$\mathbf{F}_a\left(\mathbf{n}_i, \dot{\mathbf{x}}\right) = k_1 \left(\frac{1}{i}\sum_i \mathbf{n}_i, \dot{\mathbf{x}}\right) k_2 \frac{\mathbf{x}_{\text{goal}} - \mathbf{x}}{\|\mathbf{x}_{\text{goal}} - \mathbf{x}\|} \tag{6.5}$$

An analog adaptation according to the surface normals is also used for the damping force. This modification is chosen because the attracting element does not inject energy into the system and hence damping should do this neither. The *upper right* simulation visualizes similar effects with the same range for a V-Trap type problem. The *down right* plot depicts that also larger and sharper-edged V-Traps can be avoided even when using limited range (here 5 units) and the proposed velocity angle adaptation. This was chosen as ψ ranging form 10° up to 20°. With this modification it can be observed that the point mass avoids the obstacle earlier, which is mainly due to the range of influence. Furthermore, it also leaves the objects influence earlier because of the velocity angle adaptation. If the range is further increased and the current factor K_i is not chosen too low, the AO does not even enter the local minimum part of the object. This behavior is shown in the *down left* plot.

A positive effect that was observed for all generated traps is the capture tendency that is caused by the obstacle character of CFs. This avoids losing contact while passing extrema of obstacles and thereby generates smooth trajectories for the obstacles.

Next we discuss the 3D analysis.

6.2 3D Analysis: Simulation and Experiments

After the promising 2D simulation results the implementation of CFs was extended to a 3D algorithm and the corresponding environment (as described in Sec. 5.1.2). Multiple random and quite complex scenarios and first experiments were carried out and are discussed in the following.

6.2.1 Simulation

The 3D simulations we discuss now were evaluated in a similar order as for the 2D case, as the scenarios also increase in complexity. 3D random testbeds of spheres, complex traps that are similar to the already introduced U-trap, a box type object, and first dynamic simulations with moving obstacles are described.

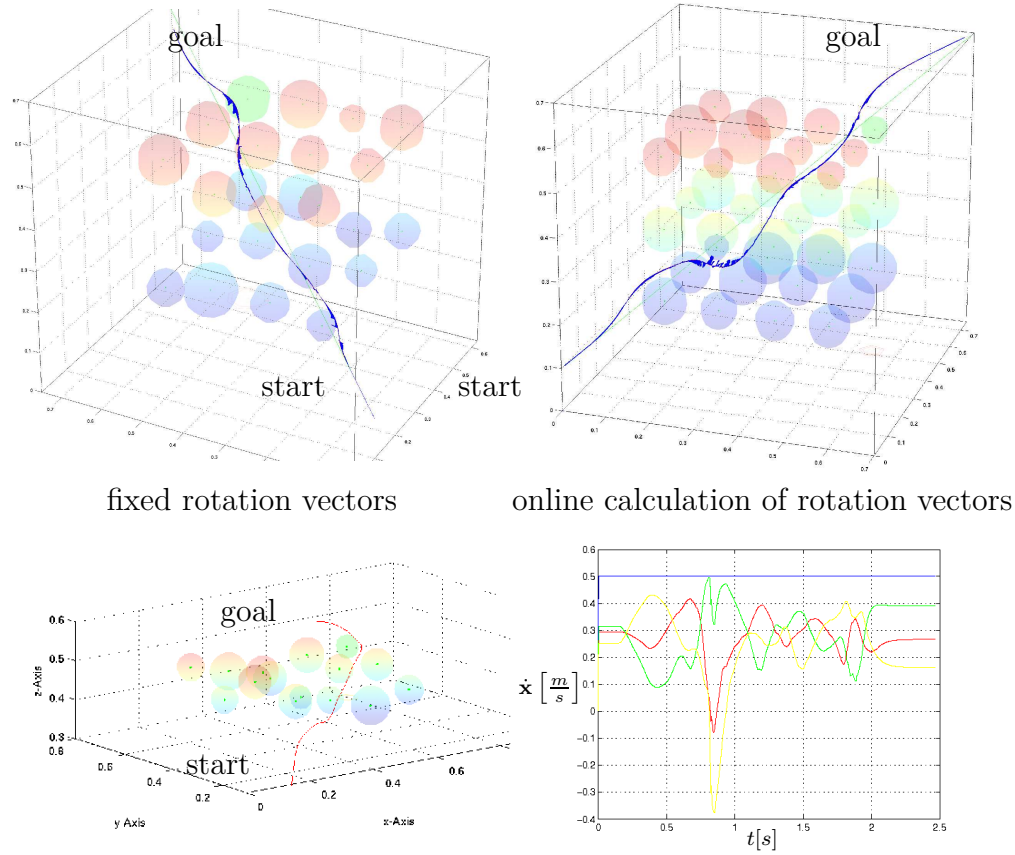


Figure 6.10: 3D avoidance of sherical objects with a point mass.

Static random testbed of spheres

The random testbed were again chosen to evaluate the general behavior of the CF approach for varying scenarios and find potential problems of the method. The sphere testbeds are generated based on a grid in which on every element a sphere with random radius is positioned. In Fig. 6.10 four sample testbeds are shown.

The obstacle influence range in these scenarios shown in Fig. 6.10 is between 0.03 and 0.05 units. The *upper left* figure depicts the behavior for fixed rotation vectors $\mathbf{r}_j$. The virtual point mass takes one of the shortest trajectories towards the goal.

The images *down left* and *top right* are simulated for calculating the rotation vectors online. The derivation of $\mathbf{r}_j$ leads to a virtual point mass

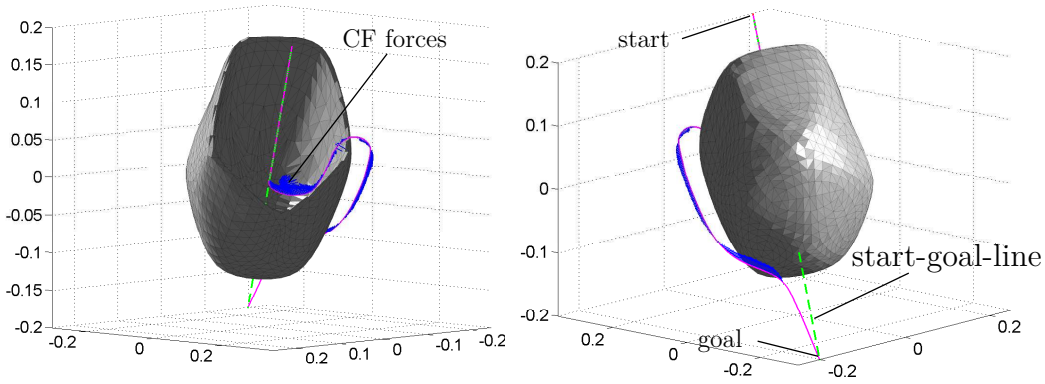


Figure 6.11: Circular field based motion generation reactively passes a dead-end.

trajectory with less risk of collide with obstacles while moving towards the goal. The trajectory is similarly smooth as the previous example and needs approximately the same number of iteration steps for goal convergence, i.e. the generated trajectory is of similar quality as for fixed $\mathbf{r}_j$. Finally, in the image *down right* an example for a resulting velocity profile is given, showing the smooth behavior during a motion.

Trap Simulation Scenario

The already shown 2D ability to handle typical trap objects that usually lead to a dead-end for commonly used motion generation methods as potential fields is now extended to the 3D case. We consider again a U-trap and an additional box trap.

CFs on U-Trap The created U-trap (see Fig. 6.11) consists of 2860 surfaces. However, the multiple force elements acting from the surface in the region of consideration can still be calculated in real-time for the virtual point mass. The sphere of influence is chosen small enough so that the virtual particle still enters the object. After the first obstacle surfaces are in range of perception the particle starts to follow the obstacle surface and finds it way out of the trap towards the goal. The used attractor is again described by Equ. 6.5 and the damping modification was applied as well.

CFs on Box-Trap To show the capabilities of CFs for a particularly difficult scenario, we have constructed a box that can only be entered via a small entrance. Fig. 6.12 depicts this box obstacle that consists of 4200 surfaces.

Again, the CF based method is able to find its way to the goal in real-time and no collisions with the object are observed.

Again the entry into the box is chosen to be larger than the perception sphere (gray) of the virtual particle. Thus, the robot enters the box, starts following the wall on the other side of the box, and is finally guided into the desired goal state.

Dynamic Objects

The behavior in a highly dynamic environment was also simulated in order to judge the behavior for this especially important situation for a robot that is acting in a possibly dynamic environment. For this the algorithm is tested for some special cases of obstacle motion where the velocity of the obstacles is approximately the same or higher than the robot velocity, see Fig. 6.13. In order to avoid the approaching obstacles we chose to implement different strategies that incorporate the current situation, differentiating whether the object approaches from behind or from the side and above. Avoiding an obstacle approaching from the front at high velocity can be performed with the basic approach we already used up to now. Therefore, we omit the discussion of this particular case: For the other cases we chose following strategies.

- Relative velocity based
- Overtaking: switch orientation of obstacle rotation vector by 180° if the relative velocity is a towards motion (AO) or towards the goal

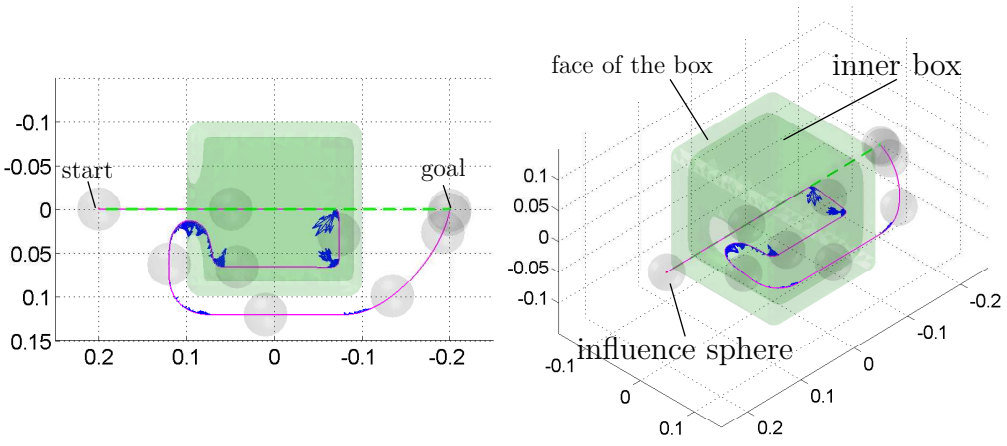


Figure 6.12: Using CFs to reactively pass a complex dead-end (top and 3D View).

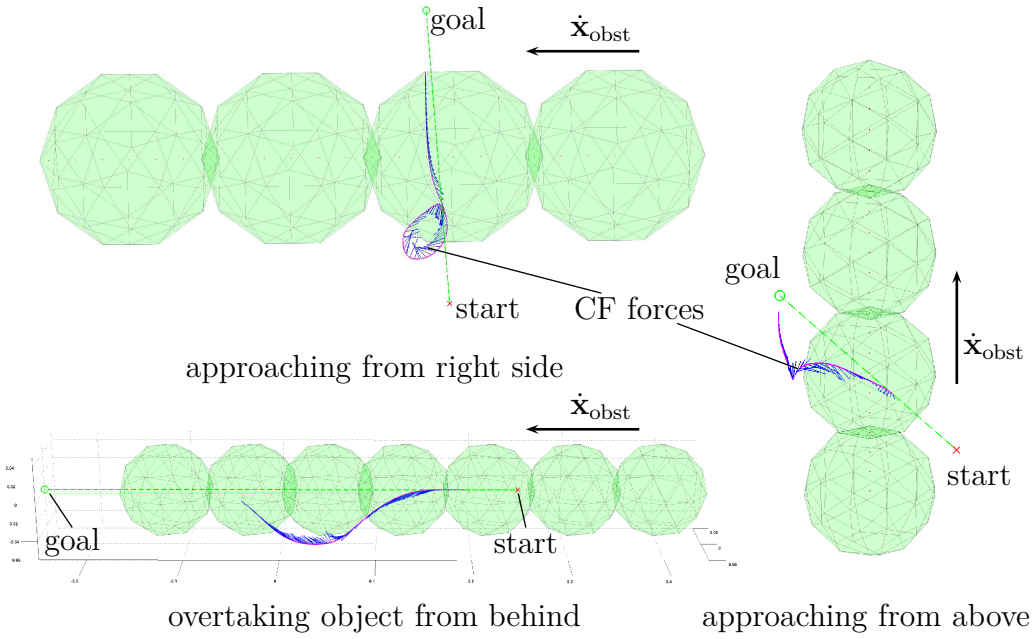


Figure 6.13: Point mass avoiding dynamically moving objects.

(different approaches possible)

- Object from side/below/above: continuous turning of obstacle rotation vector depending on the (position to) OO and the relative velocity.

An open problem we still encounter is how to consistently combine the used strategies or to find **one** continuous strategy. However, this is left for future work.

In the following the first experimental evaluation for avoiding static obstacles and a human operator are presented.

6.2.2 3D Experiments

In this subsection we shortly describe two distinct experiment that were performed to analyze the performance of the CF algorithm on the real robot setup. The LWR-III was always operated in Cartesian impedance control and the CF scheme commanded the velocity interface (attractor) of the robot by forces.

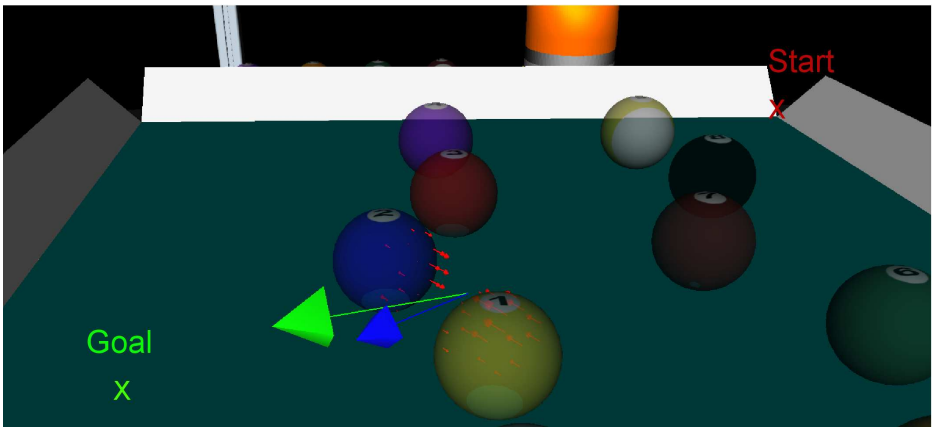


Figure 6.14: Experimental evaluation to avoid multiple static obstacles.

Avoiding multiple static Obstacles

The very first experiment performed with CFs is to circumvent a parcour of billiard balls that are statically located on a billiard table setup. The cue tip is represented by the virtual point mass and influenced by the CF forces (red arrows), see Fig. 6.14. In this figure the visualization for one particular time instant is shown. For this experiment every billiard ball is represented by a 80 triangle surface. The starting point for the motion is in the upper right corner and the goal in front left of Fig. 6.14. The billiard balls are static and once localized by the celling camera.

Passively tracked Static Obstacle Experiment

The second experiment was also performed on the Co-Worker setup. The obstacle, in this case a human whose arm is equipped with passive tracking markers is smoothly circumvented. The infrared tracking system used in this experiment was described in Sec. 4.1. The robot motion and the results are illustrated in Fig. 6.15.

Again, the robot end-effector (located in the center of the wrist sphere) is represented by a virtual point mass. The robot is smoothly guided around the obstacle, while keeping the orientation constant (3D motion with fixed rotation). The main limitation in this experiment is the quite noisy data of the tracking system when objects (in this experiment it is the robot itself) come close to the passive marker. This leads to a significant increase in recognition accuracy of the ART system.

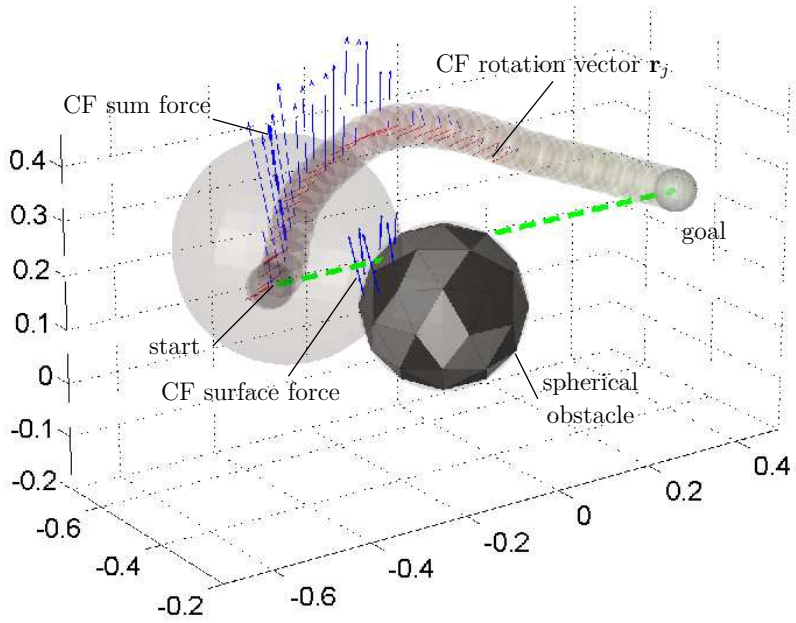


Figure 6.15: Circular fields for reactively avoiding visually tracked objects in the robots workspace.

Fig. 6.15 illustrates the influencing sphere (gray) as well as the marker position that is represented by the ball object. This obstacle is also associated with 80 surfaces (dark gray). The virtual forces acting on the obstacle surface are shown for one of the first poses of the robot. Along the past motion the resulting forces acting on the virtual particle are also visualized (blue arrows). The dashed green line is the direct connection from start (left) to goal (right).

Next, we discuss the 6D evaluation based on a multitude of simulations and experiments, leading to a complex tactile exploration experiment.

6.3 6D Analysis: Simulation & Experiments

In this section we first show results for various task related motions in simulations and experiments. Furthermore, we describe the simulation and experimental achievements for exploration and avoidance of unknown wire elements.

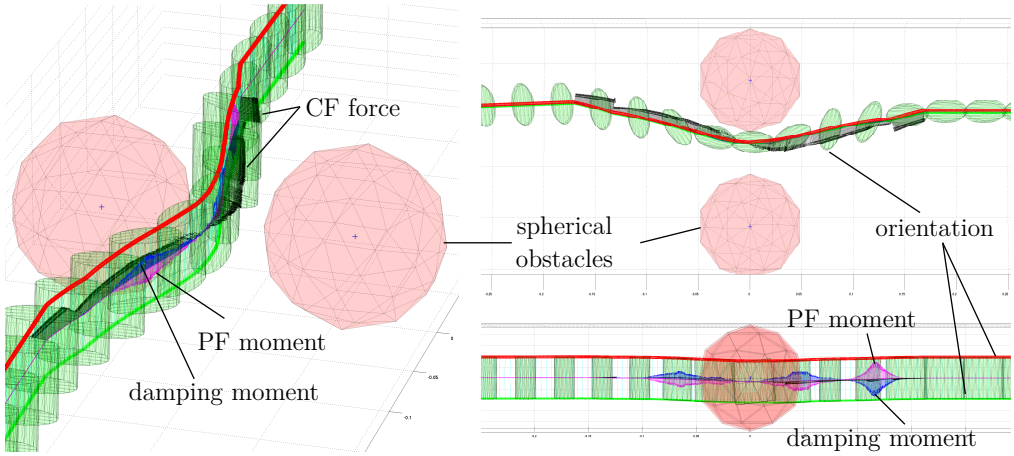


Figure 6.16: 6D ellipsoid bar moving through a narrow passage.

6.3.1 End-Effector - Simulation

In this simulation we discuss an ellipsoid bar element that is used to verify the 6D motion generation of the hybrid approach described in Sec. 3.3.3. CFs are used for translational and PFs for rotational motion generation. The behavior of the virtual end-effector in a narrow passage problem is depicted in Fig. 6.16.

The hull bodies of the ellipsoid end-effector represent the AO and the two OOs create a narrow passage problem. In Fig. 6.16 the CF forces (black), the rotational moments (magenta) resulting from PFs, the strong damping moments (blue) on the rotation, as well as the orientation (red markers on top, green markers on the base and a cyan connection line in between) are illustrated. The attracting, damping, and CF forces generate a translation motion and PFs in combination with damping moments contribute to the orientation motion. By this constellation a smooth 6D trajectory is generated that ensures also collision avoidance.

The *top right* part of the image shows that the AO does not pass the narrow passage exactly along the middle between the two OOs. The asymmetric passing is however mainly because no velocity angle adaptation was used for this scenario, which would partly compensate for this. Furthermore, there are discontinuous CF forces when entering the range of the obstacle influence.

Next, we discuss a 6D collision avoidance experiment with a dynamically moving human that is associated with a sphere and the robot end-effector with an ellipsoid.

6.3.2 6D Collision Avoidance Experiment

The following experiment was performed with some changes compared to the previous one. For the present case we use the CF shaping described in Sec. 3.3.2 and the velocity angle adaptation from Sec. 3.3.2. We also consider the relative velocity between robot and obstacle, which results in a stronger derivation of the AO if the OOs is towards the AO motion direction. Please note that due to the CF shaping we disabled the possibility to react to overtaking motions. However, if the obstacle approaches the end-effector, PF forces generate a translational movement that prevents a possible collision. The interface to the RCU consists of virtual forces that are used as a disturbance input for the attractor and the moments that act directly via the Jacobian as a virtual desired torque.

- Translation: real-time attractor with shaped velocities: input $\mathbf{f}_{\text{virt}} = \mathbf{f}_{\text{ob},n}$
- Rotation: virtual moments via Jacobian $\boldsymbol{\tau}_d = J_{EE}^T \mathbf{F}_{\text{ob},n}$ acting on Cartesian impedance control with $K_\varphi = \text{diag}\{60\}$ Nm/rad to show significant response.

The obstacle forces act directly on the dynamics of the artificial attractor, therefore, only indirectly on the Cartesian impedance control level. The moments, however, are directly incorporated into the low-level torque control scheme via the end-effector Jacobian and therefore directly generate desired torque values. The experiment was performed for two different start and final configurations. In task one the end-effector is oriented in z-direction and performs a reference motion that is almost parallel to the y-axis. In task two the end-effector is oriented closely in x-direction and performs a task that is almost in parallel to the x-axis. The resulting forces from the obstacles are depicted for such a run in Fig. 6.17.

On the left side the real posture of the OO (passive marker on hand) and the final configuration of the robot arm are illustrated. In the middle the 3D visualization before passing is shown and on the right the one while passing the dynamic obstacle. Additionally, the environment frames, their respective changes, and the progression of the obstacle forces are illustrated. The reorientation of the end-effector, which also has large impact on the motion of the forearm and secondary on the upper arm of the robot can be observed. Additionaly, Fig.6.18 depicts the Cartesian forces (left) and moments (right) that were generated during the motion. The red curves

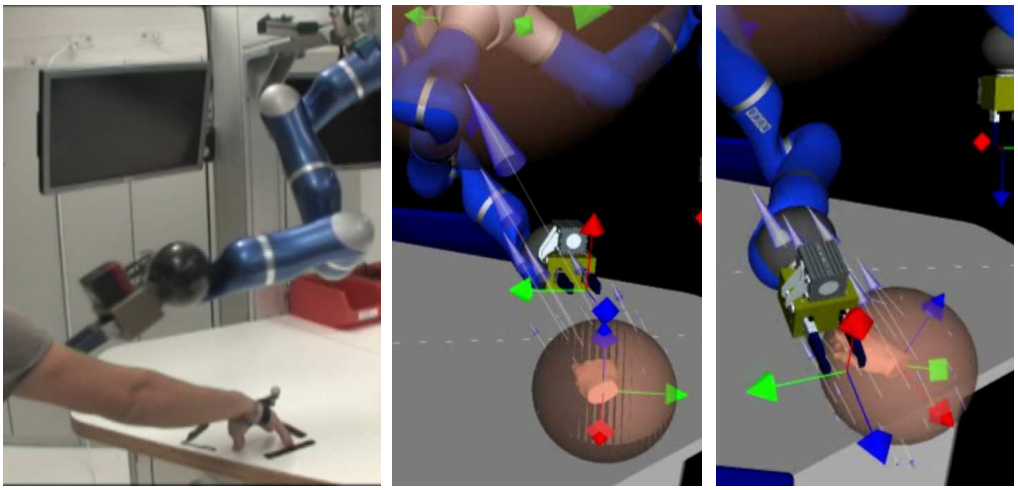


Figure 6.17: Avoidance behavior for task two. The depicted arrows denote the occurring forces calculated by the RCA.

show the absolut values, whereby the $x - y - z$ components are illustrated as green, blue and yellow curves.

Next, we describe the hot wire exploration simulation and there approach.

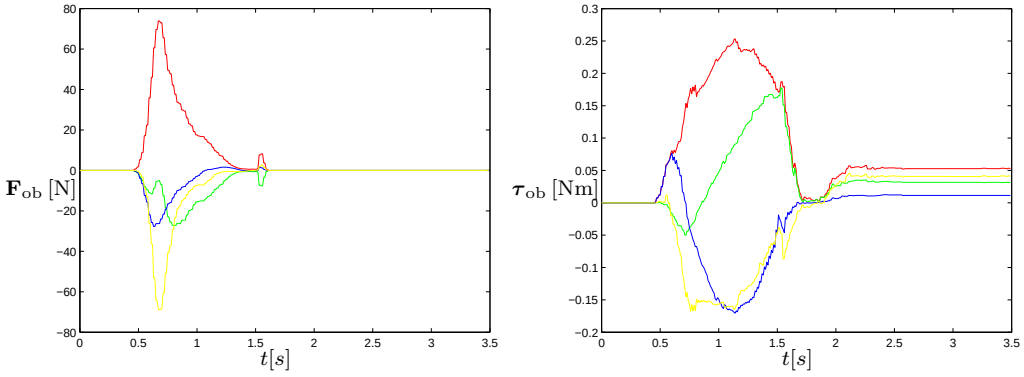


Figure 6.18: The obstacle forces (left) and moments (right) resulting on end-effector for a 6D motion.

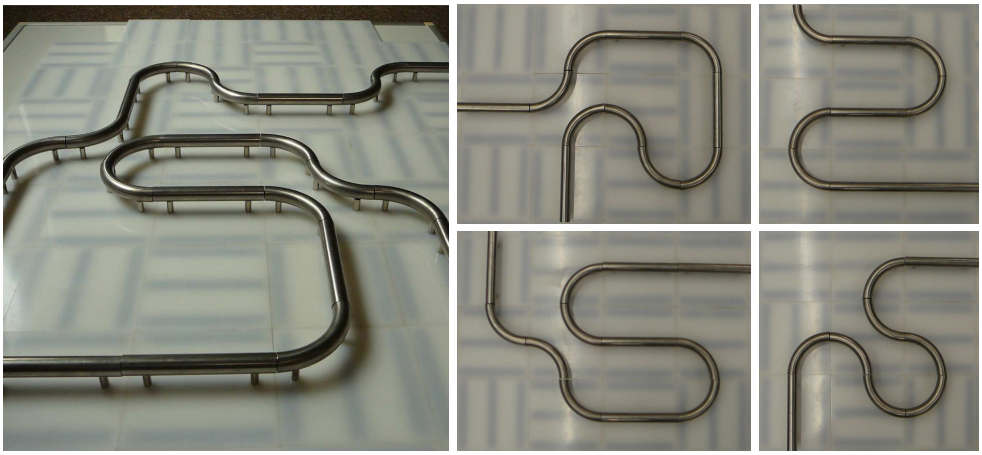


Figure 6.19: The hot wire assembly.

6.3.3 Hot Wire Exploration - Simulation

In the following subsection we discuss the application of the developed collision avoidance schemes for tactile exploration. For this we use the information about interaction forces we get from the LWR-III to generate virtual maps. These are then interpreted as an obstacle we would like to circumvent. This enables us to explore complex 6D objects and, after the exploration phase is over, to perform motions that do not collide with the obstacles anymore. The used experimental setup is shown in Fig. 6.19 by a perspective view on the 2D problem and a top view for multiple scenarios for the experiment.

The hot wire exploration simulation is intended as a show-case that it is possible to explore a wire by tactile contact information only and especially without any visual feedback. Furthermore, we use this information to built a tactile map of the object that can then be used for future motion generation.

Approach

The task to explore an object by interoceptive perception capabilities of a robot is still a major challenge in robotics. The combination of tactile exploration and local motion generation algorithms is a novel way to approach the problem. The underlying idea is to associate the robot with a virtual 6D end-effector object, which is guiding the motion in a virtual environment. This environment is incrementally built based on interaction forces sensed during motion. In our implementation we use a torus as the virtual represen-

tation of our gripper to explore a wire labyrinth that is not known a-priori, see Fig. 6.19. We generate translational and angular velocities to control the robot end-effector directly in Cartesian impedance control. This makes it possible to generate motions not only by means of interaction control but to use the sensed forces to explore a virtual space.

The basic approach is to explore the physical object (wire) by using force and position measurements of the internal sensors of the robot and to construct an "avoidance map" based on a simple assumption on the unknown object geometry. Initially, we assume the wire to be an infinite straight object with certain radius. Furthermore, we assume that the robot initial pose is aligned with the beginning of the wire-type object to be explored. Based on the sensed tactile information as e.g. force, moment, position, and orientation we re-orientate this wire element incrementally. The a-priori as-

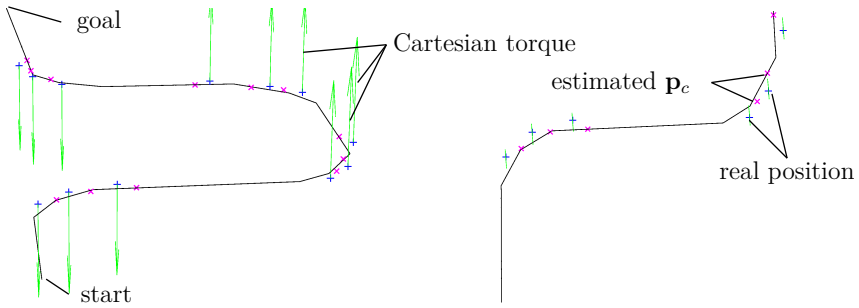


Figure 6.20: Hot wire generation sketch.

sumption on the orientation $\mathbf{x}_{\text{base0}}$ of the wire element is used to rotate the basic wire element object into the respective start pose.

For calculation the rotation vector based on sensor input two approaches can be used. First, we may assume that sensed forces are directly aligned with the geometric normal of the object. This leads to aligning new elements orthogonally to this vector. Secondly, the rotation can be performed according to the measured moment or torque (Fig. 6.20 green arrows). According to some initial testing, the moment or torque approach appeared to be the more robust approach due to non negligible friction effects during contact. Furthermore, due to the absence of an external force/torque sensor we calculate the forces and moments from an estimation of the external joint torques $\boldsymbol{\tau}_{\text{ext}}$.

$$\mathbf{x}_{\text{baseNew}} = \mathbf{R}(f(\boldsymbol{\tau}_{\text{ext},7}))\mathbf{x}_{\text{base0}}, \tag{6.6}$$

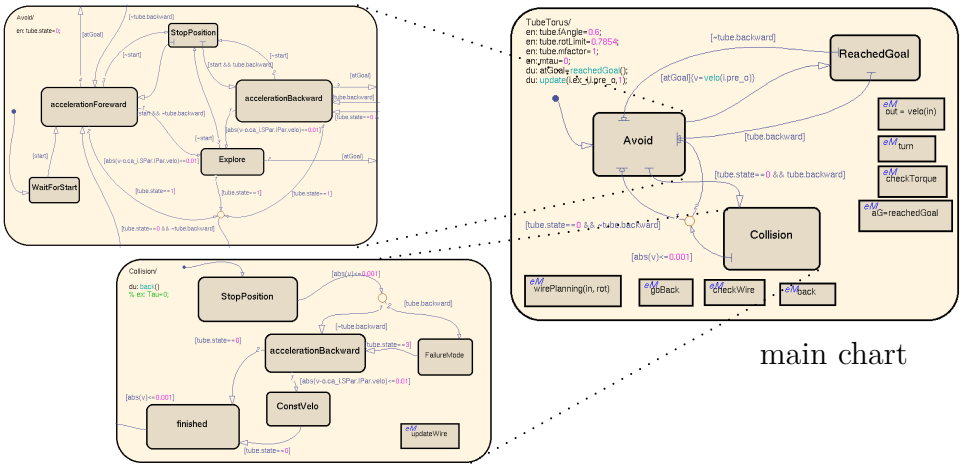


Figure 6.21: Implementation of the hot wire State Flow planner.

where f is a suitable function to limit the calculated rotation for high torques $\boldsymbol{\tau}_{\text{ext},7}$ on the seventh joint. $\mathbf{x}_{\text{baseNew}}$ is the re-orientated wire direction. The re-orientation is then performed at the estimated wire center point $\mathbf{p}_c$ of the contact (Fig. 6.20 magenta marks). This is obtained by the actual position of the torus $\mathbf{x}_{\text{torus}}$ (blue marks) and the external torque $\boldsymbol{\tau}_{\text{ext},7}$.

$$\mathbf{p}_c = R_{\text{torus}} \frac{\boldsymbol{\tau}_{\text{ext},7} \times \mathbf{x}_{\text{base0}}}{\left\| \boldsymbol{\tau}_{\text{ext},7} \times \mathbf{x}_{\text{base0}} \right\|} + \mathbf{x}_{\text{torus}} - r_{\text{torus}} r_{\text{wire}} \frac{\boldsymbol{\tau}_{\text{ext},7} \times \mathbf{x}_{\text{baseRot}}}{\left\| \boldsymbol{\tau}_{\text{ext},7} \times \mathbf{x}_{\text{baseRot}} \right\|} \quad (6.7)$$

In the following $\mathbf{p}_c$ and the new orientation $\mathbf{x}_{\text{baseRot}}$ are used to calculate the crossing point with the already existing wire element to redefine the internal model of the wire that is a polygon of line elements (Fig. 6.20 black line).

The motion generation approach is to use only CFs (no damper and attractor), thus a free floating mass for translation and PFs with damping and no attractor for rotation (rotation energy is always decreasing: $D \propto E_{\text{rot}}$). There is no "long-term" rotational local minimum due to map building.

The Simulink State Flow Implementation

In State Flow a reactive planner is implemented to control the order of action during the wire exploration task. The main chart is shown in Fig. 6.21. The main chart consists of three sub-states:

1. the "avoid" state,
2. the "in collision" state,

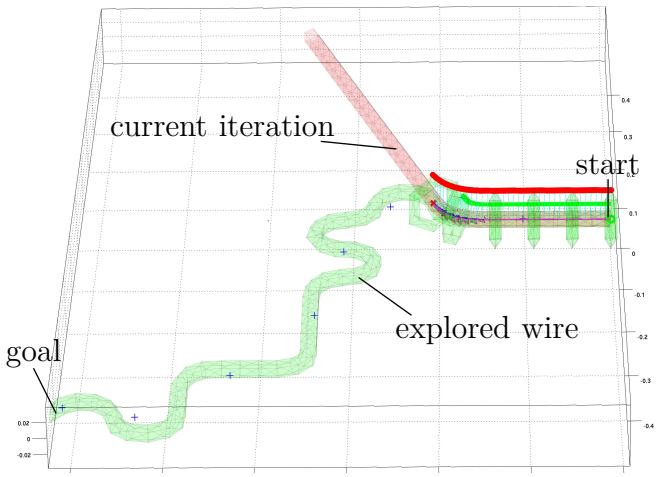


Figure 6.22: Simulation of the hot wire exploration: started.

3. and the "reached goal" state,

where the end-effector is stopped and waits for new commands. The "avoid" state consists of 5 sub-sub-states that are self-explanatory, the "wait for start", "stop at this position", "acceleration forward", "acceleration backward", and the "explore" state. The latter follows the current internal wire model and reacts if external forces become too large (leads to a change into the sub-state "in collision").

The "in collision" determines the course of action when being in collision: The end-effector is stopped and the contact forces observed for creating a new internal model. Then, by calculating the crossing point of the new line element the new destination for retraction is calculated. The avoidance map is not updated while moving back in order to avoid discontinuous forces. By the "acceleration backward" and "constant velocity" sub-sub-states the robot is then moving back. When finished, the motion is stopped again. Now, the avoidance map is updated and the state change to the sub-state "avoid" is executed.

With the described behavior it is possible to fully explore the wire and afterwards navigate with the robot through the wire without causing any collisions anymore.

The Simulation

In this subsection we show via simulation the feasibility of the approach. For this we implemented two independent wires, see Fig. 6.22 and Fig. 6.23.

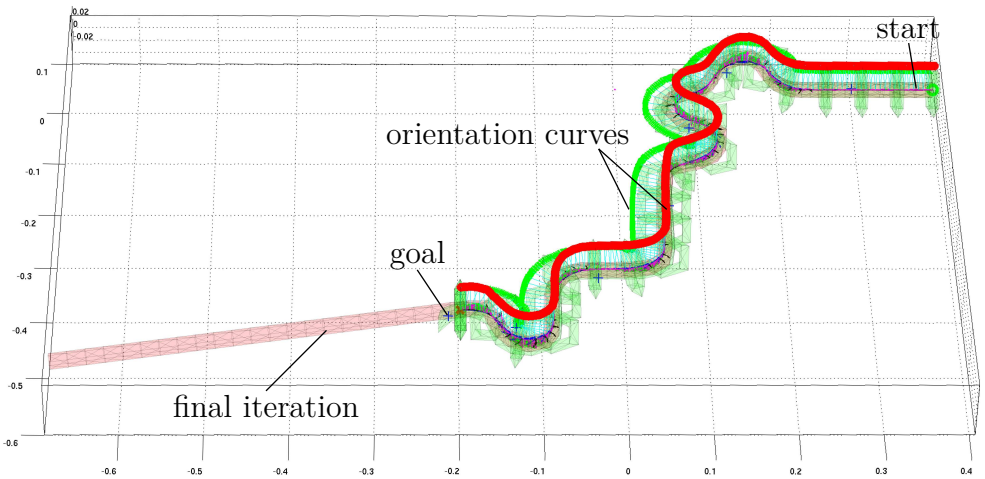


Figure 6.23: Simulation hot wire exploration: finished.

A randomly generated wire (green) shall be explored with a ring torus (green) by continuously deforming an initial wire (red). The orientation of the ring torus is also marked with red and green marks. For both wire examples the resulting torques on the torus are calculated and this residual is integrated until a certain threshold is surpassed. If this incident occurs, the current difference of moments is associated with a contact moment and used to re-orientate the wire (red). As shown in Fig. 6.23 the entire wire can be explored such that the entire geometry of the object can be reconstructed.

Next, we discuss the experimental implementation, which also shows the quality of the algorithm.

6.3.4 Hot Wire Exploration - Experiment

The hot wire experiment is performed with the same approach described for simulation. The only difference is that the model of the collision avoidance simulation is of course not equal to the real wire. Therefore, we include also some measured robot states in order to be able to observe real world states. The control-loop is closed via the desired model of the robot. This means that the collision avoidance is used to command only the desired velocity of the robot by generating reference translational and angular velocities. However, if a contact between gripper and the real wire occurs, we use the real state of the robot to generate the avoidance map as accurate as possible. The contact treatment is left to the local behavior of the impedance controller and sensing of external forces. In Fig. 6.24 the results of one of the hot wire

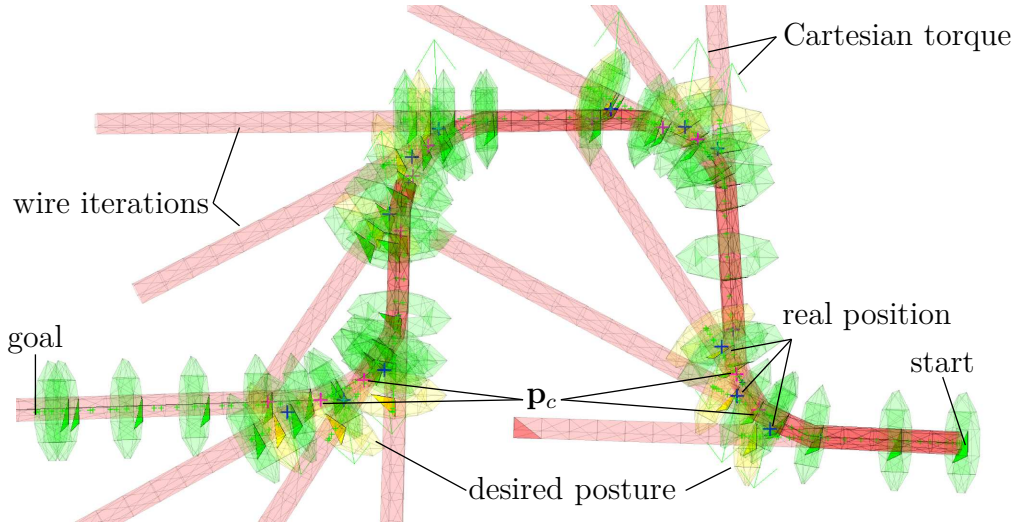


Figure 6.24: Perspective view on the explored avoidance map and intermediate steps.

experiments is depicted.

The virtual wire element is re-orientated at every contact point with the real wire. At the time instant contact is detected the desired state is illustrated by a yellow torus. The real robot position is marked with a blue cross mark. The difference between real and desired state of the robot can be observed by the position difference. Magenta crosses mark the points of the estimated position of the wire center during contact. For the contact incidents also the contact torques (green arrows) are depicted (visible for the two left turns of the wire).

This experiment was performed with multiple wire configurations, see Fig 6.19, which were also changed online. Examples of such generated wire exploration maps are shown in Fig. 6.25. On the left side we can see the 2D movement (a projection of the true motion) for a quite complex wire scenario. The dead-end movement shows the contact behavior with the wire. Also the red lines for exploring the wire are not perfectly aligned with the backward motion of the torus which is indeed smoother. On the right hand side the force measurements are shown, which are clearly not normal with respect to the imaginable real wire. This observation justifies the chosen approach without needing to estimate fiction properties.

Overall this chapter showed numerous show cases that collision avoidance while the robot is interacting with its environment is a task that can not be handle by a single algorithm. There are needed, reactive planner for high-

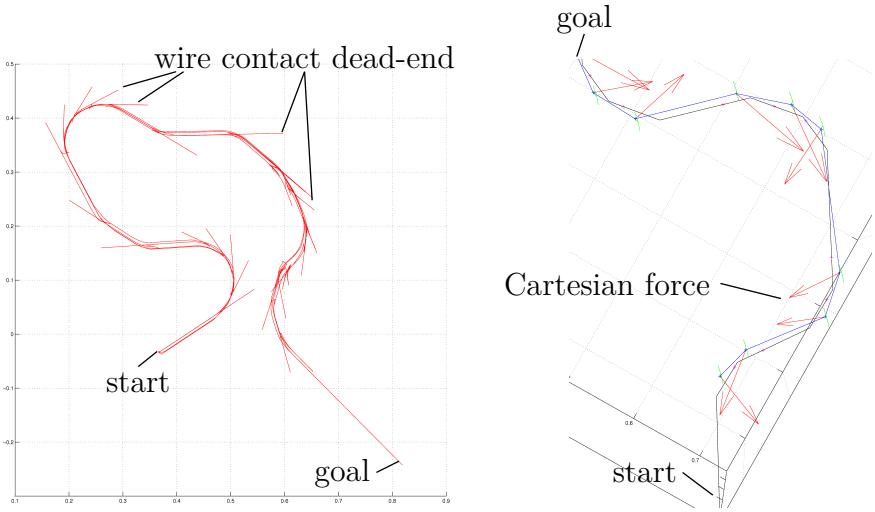


Figure 6.25: Hot wire experiment, motion and force-moments.

level decision, algorithms for translation and algorithms for a robust rotation behavior as well as proper controller as e.g. Cartesian impedance control. It was shown that the implemented RCA is a platform that can handle this quite complex task.

Next it is given a conclusion and a outlook for future developments.

7 Conclusion and Outlook

In this thesis we analyzed existing collision avoidance schemes and discussed their applicability to articulated manipulators in dynamically changing environments for applications in physical Human-Robot Interaction. We developed an extensive simulation environment with test scenarios of rising complexity in 2D, 3D, and 6D. After reviewing the state-of-the-art and excluding several motion generation algorithms based on objective criteria, several selected algorithms were evaluated and compared in rather complex 2D simulations. These evaluations enabled us to focus on the particularly well suited algorithms. After significantly extending these schemes, such that we are able to use them in the modular state-based control architecture of the DLR LWR-III, we tested the algorithms and strategies in a first 3D experiment. The performed experiments proved the capabilities of some preliminary algorithm designs in an experimental test scenario in which the task was to circumvent a static human. We further extended the original definition of CFs and finally developed a hybrid CF-PF approach for 6D reactive operational space real-time motion. Following the system requirement generated by the structured analysis, a 6D MATLAB/Simulink/StateFlow implementation was created to perform various experiments. We analyzed collision avoidance for static multi-object parcours and were able to show the performance for avoiding dynamically moving humans. Furthermore, we developed a powerful algorithm for performing tactile exploration of complex planar 3D wire elements, whose structure is a-priori unknown. All stated problems were successfully solved and showcase the effectiveness of the designed algorithms. We created also a flexible and easily extendable software structure with variable and online controllable input and output interfaces for continuing this line of research in the future. This brings us to the outlook of the thesis.

The present implementation offers multiple options of expansion and optimization for the near future. Currently, the RCA is still running within the MATLAB engine because the initial generation of objects is still integrated into the RCA software structure, therefore increasing the calculation load. At the current stage this still prevents the full implementation on a real-time system. However, it is straight forward to remove the loading of

object data from the actual algorithmic part e.g. via a ring-buffer loading interface. This will presumably enable a hard real-time implementation of the algorithm, which will speed up the RCA into the range of at least some milliseconds. Furthermore, the robot objects surfaces can be extended by using more accurate continuous meshes as e.g. the ones described in [57] and therefore, achieving a more suitable geometric hull of the true robot structure. Also the extension of the proposed algorithms to more than a single robot is subject to contemporary work. This will also enable not only avoiding external obstacles in a more complex multi-robot scenario but also the prevention of the robots with each other.

Since objects are fully described by their surface normals, associated with the according values for the surface size, we intend to assign varying gains. They are sought in particular for representing the potential danger that the respective part of the real robot is emanating. This would lead to a more drastic response to human proximity for these special robot elements.

For acceleration of the overall calculation time we pointed out the high degree of potential parallelizability of the developed algorithms. We expect that utilizing multi-core-implementations e.g. on GPUs are very advantageous and may vastly decrease the calculation time. Apart from these implementation aspects our further interest lies in gaining more theoretical insight into the stability and goal convergence properties of the novel schemes. Apart from this system theoretic result we would also like to elaborate the immanent runtime characteristics in terms of $O(.)$.

As a further potential candidate for real-time collision avoidance we considered also Harmonic Potential Fields that were left out for brevity. We intend to investigate and compare them with our enhancement of CFs in the future.

Finally, the incorporation of obstacle dynamics for the definition of the CF current could be advantageous and will supposedly considered in the future.

To sum up, the final goal of this line of research is to provide a deterministic, fast, and variable algorithmic foundation that is also theoretically sound. The implemented system shall be able to fully exploit the capabilities of the entire Co-Worker scenario and is however still easily expendable for other applications.

Bibliography

[1] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger. The DLR lightweight robot lightweight design and soft robotics control concepts for robots in human environments. *Int. J. Rob. Res.*, 34(5):376–385, 2007.

[2] A. Albu-Schäffer, C. Ott, and G. Hirzinger. A passivity based cartesian impedance controller for flexible joint robots - part ii: Full state feedback, impedance design and experiments. In *IEEE International Conference on Robotics and Automation*, pages 2666–2672, 2004.

[3] A. Albu-Schäffer, C. Ott, and G. Hirzinger. A unified passivity-based control framework for position, torque and impedance control of flexible joint robots. *Int. J. Rob. Res.*, 26(1):23–39, 2007.

[4] Ascending Technologies GmbH. Technik. Available online at `http://www.asctec.de/technik/`, August 2010.

[5] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12:43–77, 1994.

[6] A. Bierbaum, M. Rambow, T. Asfour, and R. Dillmann. Grasp affordances from multi-fingered tactile exploration using dynamic potential fields. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 168 – 174, Paris, France, 2009.

[7] M. Brady. Artificial intelligence and robotics. *Artifial Intelligence*, 26:79–121, 1985.

[8] O. Brock and O. Khatib. Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths. In *IEEE Int. Conf. on Robotics and Automation*, pages 550–555, 2000.

[9] C. Brom. Hierarchical reactive planning: Where is its limit? *MNAS workshop*, 2005.

[10] J. Chestnutt, J. Kuffner, and T. Kanade. Vision-guided humanoid footstep planning for dynamic environments. In *Proc. of the IEEE-RAS/RSJ Int. Conf. on Humanoid Robots*, pages 13–18, 2005.

[11] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[12] J. Czarske, T. Pfister, C. Bayer, and H. Leuterer. Messtechnik (Mess- und Sensortechnik) - Vorlesungsskript - TU-Dresden, 2006.

[13] D. Ebert and D. Henrich. Safe human-robot-cooperation: Image-based collision detection for industrial robots. In *In: IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1826–1831, 2002.

[14] M. S. for Ahmad Masoud. Harmonic potential field motion planning - homework 8. *King Fahd University of Petroleum and Minerals*, 2009.

[15] U. Frese and H. Täubig. Modelling and calibration technique of laser triangulation sensors for integration in robot arms and articulated arm coordinate measuring machines. *Research Report RR-09-01*, 2009.

[16] M. Fuchs, C. Borst, P. R. Giordano, A. Baumann, E. Krämer, J. Langwald, R. Gruber, N. Seitz, G. Plank, K. Kunze, R. Burger, F. Schmidt, T. Wimböck, and G. Hirzinger. Rollin' justin - design considerations and realization of a mobile platform for a humanoid upper body. In *IEEE International Conference on Robotics and Automation*, pages 4131–4137, 2009.

[17] T. Gecks and D. Henrich. Path planning and execution in fast-changing environments with known and unknown obstacles. In *International Conference on Intelligent Robots and Systems in San Diego, USA*, pages 21 – 26, 2007.

[18] W. E. Green and P. Y. Oh. Optic flow based collision avoidance on a hybrid MAV. `prism2.mem.drexel.edu/.../finalGreenRas2006.pdf`, 2006.

[19] S. Haddadin, A. Albu-Schäffer, G. Hirzinger, and A. D. Luca. Collision detection and reaction: a contribution to safe physical human-robot interaction. In *Proceedings and presented at IROS 2008*, pages 3356 – 3363, 2008.

[20] S. Haddadin, S. Parusel, R. Belder, J. Vogel, T. Rokahr, A. Albu-Schäffer, and G. Hirzinger. Holistic design and analysis for the human-friendly robotic co-worker. *International Conference on Intelligent Robots and Systems*, 2010.

[21] S. Haddadin, H. Urbanek, S. Parusel, D. Burschka, J. Roßmann, A. Albu-Schäffer, and G. Hirzinger. Real-time reactive motion generation based on variable attractor dynamics and shaped velocities. In *In Proceedings of the IEEE International conference on Intelligent robots and systems*, 2010.

[22] S. Haidacher, J. Butterfass, M. Fischer, M. Grebenstein, K. Joehl, K. Kunze, M. Nickl, N. Seitz, and G. Hirzinger. DLR Hand II: Hard- and software architecture for information processing. In *IEEE International Conference on Robotics and Automation*, pages 684 – 689.

[23] G. Hirzinger, N. Sporer, M. Schedl, J. Butterfaß, and M. Grebenstein. Torque-controlled lightweight arms and articulated hands: Dowe reach technological limits now? In *The International Journal of Robotics Research*, volume 23, pages 331–340. SAGE Publications, April-May 2004.

[24] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[25] T. Hoshi and H. Shinoda. A sensitive skin based on touch-area-evaluating tactile elements. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, page 15, Washington, DC, USA, 2006. IEEE Computer Society.

[26] K. Hsiao, L. Kaelbling, and T. Lozano-Perez. Task-driven tactile exploration. In *Robotics: Science and Systems*, Zaragoza, Spain, 2010.

[27] W. H. Huang, B. R. Fajen, J. R. Fink, and W. H. Warren. Visual navigation and obstacle avoidance using a steering potential function. *Robotics and Autonomous Systems*, 54(4):288–299, 2006.

[28] D. Isla. Handling complexity in the Halo 2 AI. *Gamasutra*, March 2005.

[29] K. Janschek. *Systementwurf mechatronischer Systeme*. Springer, 2010.

[30] J. Kim and P. K. Khosla. Real-time obstacle avoidance using harmonic potential functions. pages 790 – 796, 1991.

[31] D. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *IEEE International Conference on Robotics and Automation*, pages 1–6, April 1987.

[32] D. Koditschek. Exact robot navigation using cost functions: the case of distinct spherical boundaries. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 1791–1796, 1988.

[33] J. Kuffner Jr. and S. M. Lavalle. RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 995–1001, 2000.

[34] D. Kulić and E. Croft. Pre-collision safety strategies for human-robot interaction. *Auton. Robots*, 22(2):149–164, 2007.

[35] S. M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Department of Computer Science Iowa State University, 1998.

[36] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[37] Lemonodor. weblog by John Wiseman, Anti-UAV UAV. Available online at `http://lemonodor.com/images/peregrine-anti-uav-s.jpg`, August 2010.

[38] A. D. Luca, A. Albu-Schäffer, S. Haddadin, and G. Hirzinger. Collision detection and safe reaction with the DLR-III lightweight manipulator arm. pages 1623–1630, 2006.

[39] A. D. Luca and W. J. Book. Robots with flexible elements. In *Springer Handbook of Robotics*, pages 287–319. Springer, 2008.

[40] A. D. Luca and L. Ferrajoli. Exploiting robot redundancy in collision detection and reaction. *International Conference on Intelligent Robots and Systems*, pages 3299–3305, 2008.

[41] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision (ijcai). In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pages 674–679, April 1981.

[42] G. F. Marcus. *The Algebraic Mind: Integrating Connectionism and Cognitive Science (Learning, Development, and Conceptual Change)*. The MIT Press, 2001.

[43] A. A. Masoud. Kinodynamic motion planning. *IEEE Robotics & Automation Magazine*, 17:85–99, March 2010.

[44] S. May, S. Fuchs, D. Droeschel, D. Holz, and A. Nüchter. Robust 3D-mapping with time-of-flight cameras. In *IEEE/RSJ International conference on Intelligent Robots and Systems*, pages 1673–1678, 2009.

[45] J. oh Kim and P. K. Khosla. Real-time obstacle avoidance using harmonic potential functions, June 1992.

[46] C. Ott. *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*. Springer Publishing Company, Incorporated, 2008.

[47] Parrot GmbH. Parrot AR.Drone the flying video game. Available online at `http://ardrone.parrot.com/parrot-ar-drone/`, August 2010.

[48] E. M. Petriu. Sensor-based robot control. Available online at `http://www.site.uottawa.ca/~petriu/`, July 2010.

[49] S. Pieper. Sensoren und Aktoren von autonomen Robotern. `http://www.uni-muenster.de/.../sensoren_aktoren_roboter.pdf`, 2007.

[50] A. Saffiotti, P. Ruspini, and W. Pedrycz. Fuzzy logic in autonomous robot navigation: - a case study, in handbook of fuzzy computation, 1998.

[51] J. Santolaria, D. G. C. Cajal, J. A. Albajez, and J. J. Aguilar. Modelling and calibration technique of laser triangulation sensors for integration in robot arms and articulated arm coordinate measuring machines. *Sensors*, 9:7374–7396, 2009.

[52] B. Siciliano and O. Khatib, editors. *Springer Handbook of Robotics*, chapter Motion Planning and Obstacle Avoidance. Springer, Berlin, Heidelberg, 2008.

[53] L. Singh, H. Stephanou, and J. Wen. Real-time robot motion control with circulatory fields. pages 2737–2742, 1996.

[54] E. A. Sisbot, L. F. Marin-urias, R. Alami, and T. Siméon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874–883, October 2007.

[55] M. Sonka, V. Hlavac, and R. Boyle. *Processing, Analysis, and Machine Vision*. Chapman & Hall, 2 edition, 1998.

[56] M. W. Spong, S. Hutchinson, and M. Vidyasagar, editors. *Robot Modeling and Control*, chapter Path and Trajectory Planning. Wiley, 2006.

[57] K. H. Strobl, E. Mair, T. Bodenmüller, S. Kielhöfer, W. Sepp, M. Suppa, D. Burschka, and G. Hirzinger. The self-referenced DLR 3D-modeler. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 21–28, 2009.

[58] A. V. Technology. Digital cameras for machine vision and life science - allied vision technologies. Available online at `http://www.alliedvisiontec.com/`, August 2010.

[59] W. T. Townsend and J. A. Guertin. Teleoperator slave - WAM design methodology. *Int. J. Rob. Res.*, 26(3):167–177, 1999.

[60] S. Wang, J. Bao, and Y. Fu. Real-time motion planning for robot manipulators in unknown environments using infrared sensors. *Robotica*, 25(2):201–211, 2007.

Selbstständigkeitserklärung

Hiermit erkläre ich, Rico Belder, geboren am 07.10.1983 in Lutherstadt Wittenberg, dass ich die von mir am heutigen Tag dem Prüfungsausschuss der Fakultät Elektrotechnik und Informationstechnik eingereichte Diplomarbeit zum Thema

Reaktive Kollisionsvermeidungsstrategien für Roboter in der direkten Mensch-Roboter Interaktion

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Dresden, den 10.09.2010 .....

Unterschrift