Safe navigation on vertical concrete structures is still a great challenge for mobile climbing robots. Although there is a large field of application it is hard to find the optimum of applicability and safety. Applicability comes with demands for fast navigation speed, high maneuverability, easy handling by the user, and high payload (e.g. inspection sensors, tools), whereas safety requires a more defensive system behavior.

This thesis addresses the problem of safe navigation in the range of wall-climbing robots. So far, closed-loop controllers for adhesion and locomotion are not sufficient to avoid a drop-off in certain situations. Therefore, additional measures are needed to improve system’s safety.

For adhesion safety a behavior-based network has been developed combining closed-loop control behaviors and deliberative components. After all, the adhesion control network is analyzed online to determine the current state of the system via an evaluation function, which is optimized using a genetic algorithm and training examples. The functionality of the developed approaches and the benefit for safety is proven in real-world experiments as well as in a simulated environment. It is shown that the prototypic robot is able to detect and avoid risky surfaces and that navigation safety could be improved tremendously.
Safe Navigation of a Wall-Climbing Robot
– Risk Assessment and Control Methods

Daniel Schmidt

Vom Fachbereich Informatik der
Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation.
Zur Begutachtung eingereicht am: 15. Oktober 2012

Vorsitzender: Prof. Dr. Katharina Zweig
Erster Berichterstatter: Prof. Dr. Karsten Berns
Zweiter Berichterstatter: Prof. Dr. Peter Liggesmeyer
Dekan: Prof. Dr. Arnd Poetzsch-Heffter

Zeichen der TU im Bibliotheksverkehr: D 386
Preface

Of course this doctoral thesis would not have been possible without the support of several other people. Whether by scientific discussions, by mechanical constructions, by supply with materials, or by giving me encouragement in the work I do – support has many faces. First of all I would like to thank my supervisor Prof. Dr. Karsten Berns. He gave me the possibility to research in the very interesting and multifaceted field of robotics. His far-sighted vision was the ideal motivation for the work on climbing robots to bring them into life. During endless discussions the scientific aspects became more and more detailed and a significant shape. Furthermore, I would like to thank my second supervisor Prof. Dr. Peter Liggesmeyer for spending his time and sharing his knowledge from the field of safety analysis and Prof. Dr. Katharina Zweig for her chairmanship during the defense of my thesis.

As a member of the Robotics Research Lab I was of course not sitting in an isolated cellar room. The work on shared hard- and software as well as on common projects led to an inspiring and constructive atmosphere with many discussions, new ideas, problems and solutions. Therefore, I would like to thank my current or former colleagues Christopher Armbrust, Michael Arndt, Sebastian Blank, Tim Braun, Patrick Fleischmann, Tobias Föhst, Carsten Hillenbrand, Jochen Hirth, Lisa Kiekbusch, Jan Koch, Helge Lerch, Tobias Luksch, Syed Atif Mehdi, Thomas Pfister, Martin Proetzsch, Max Reichardt, Alexander Renner, Daniel “Bucket Excavator” Schmidt, Norbert Schmitz, Steffen Schütz, Thomas Wahl, Jens Wettach, Jie Zhao, and Gregor Zolynski for their support in terms of countless and valuable discussions, the judgement of new ideas, server administration, proof reading and much more. I also want to express my gratitude to our secretary Rita Broschart for her administrative support in the daily concerns.

Further acknowledgements go to those colleagues who contributed to design, construction and maintenance of the real robot prototype: At first I have to thank Carsten Hillenbrand as one of the intellectual fathers of cromsci, whose preliminary work in the range of climbing systems was a starting point of my thesis. Further thanks go to Steffen Schütz for supporting me in the last phase of my work and for the help with the creation of a new prototype. In addition, I would like to thank our technician Lothar Gauß who put a lot of effort into constructive work and the organization of materials, special tools, and parts from the workshop. Their help established the prerequisites for realizing the presented work on the real prototype.

An important contribution to this thesis was also made by a couple of students. Here, I especially would like to thank Marcel Jung and Christian Marx who worked on the cromsci project with great enthusiasm. Additional thanks go to Steffen Hemer and
Anne Kickton for their support in the range of school projects. This helped me a lot to get more time for my research beside the increasing school activities of the working group. Finally, this work would not have been possible without my friends and family. Thanks go to Norbert who became a very good friend over the last ten years. It was a great time to study and to work with you. I also want to express my gratitude to my parents Silvia and Udo for supporting me during my hole time of studying and working in Kaiserslautern. They did never give up hope that I will finish this work. Thank you for believing in me. Last but not least, my thanks and love go to my wife Sarah und my little daughter Martha who gave me invaluable encouragement and supported me for all intents and purposes. This work is for you.
Abstract

Safe navigation on vertical concrete structures is still a great challenge for mobile climbing robots. Although there is a large field of application for such systems (e.g. inspection tasks, maintenance or construction) there are still no commercial robots available. The main problem is to find the optimum of applicability and safety. Applicability comes with demands for fast navigation speed, high maneuverability, easy handling by the user and high payload in terms of inspection sensors or tools for maintenance. In contrast to that, safety requires a more defensive system behavior to ensure the adhesion even under worst conditions to avoid injuries of the technical staff or damages at the system.

This thesis addresses the problem of safe navigation in the range of wall-climbing robots using negative pressure adhesion in combination with a drive system. Although such systems need to be equipped with low-level control elements for balancing the adhesion force, these closed-loop controllers are not sufficient to avoid a drop-off in certain situations. Therefore, additional measures are needed to improve the system’s safety. First of all, the different hazards affecting the robot have to be examined. Especially robot tilt and robot slip need to be handled since they are the most dangerous incidents. Based on a fault tree analysis several points of action are identified to increase the robot’s safety. The existing control components of a prototypic climbing robot are extended by additional safety measures to enhance locomotion, to ensure adhesion and to reduce risks. An advanced motion control system has been developed combining several innovative methods in the range of climbing robots such as a traction control system or a closed-loop control to minimize shear forces. These measures significantly improve robot safety during locomotion.

The basis of safe adhesion lies in the control elements of the negative pressure system. In contrast to known climbing robots a novel behavior-based network has been developed combining closed-loop control behaviors and deliberative components. After all, the structure and meta values of this adhesion control network are used to analyze the current state of the system. An evaluation function, which is optimized via a genetic algorithm based on training examples, allows an online prediction of upcoming risks leading to a drop-off. These risks can be caused by small surface irregularities and rough patches of the concrete ground and can neither be described sufficiently nor detected beforehand since there exists no suitable sensor system for this special application. Finally, corresponding counteractive measures are applied to prevent the robot from a drop-off. The functionality of the developed approaches and the benefit for safety is proven in real-world experiments as well as in a simulated environment. It is shown that the prototypic robot is able to detect and avoid risky patches and obstacles and that navigation safety could be improved tremendously.
# Contents

1 Introduction

1.1 Problem Description .................................................. 2
1.2 Objectives .................................................................. 4
1.3 Outline ................................................................... 6

2 Fundamentals

2.1 Climbing Robot CROMSCI ............................................... 9
  2.1.1 Components for Locomotion ..................................... 11
  2.1.2 Drive Kinematics .................................................. 14
  2.1.3 Adhesion System .................................................. 18
  2.1.4 Inspection and Manipulation System ........................... 24
  2.1.5 Environmental Sensors .......................................... 25

2.2 Robot Control Software .................................................. 26
  2.2.1 Behavior-based Control Architecture iB2C ...................... 27
  2.2.2 Behavior Network for Motion Control ......................... 30

2.3 Related Work in Wall-Climbing Robots ............................ 31
  2.3.1 Adhesion Principles and Locomotion ......................... 32
  2.3.2 Comparison and Discussion ..................................... 38

3 Hazards, Risks and Safety Aspects ................................. 41

3.1 Definitions ................................................................ 41
3.2 Safety Guidelines .......................................................... 44
  3.2.1 Safety Requirements in Industrial Environments .......... 45
  3.2.2 Safety Guidelines for Mobile Robots .......................... 46
  3.2.3 Self-Safety for Autonomous Robots .......................... 47

3.3 Safety Analysis and Risk Assessment .............................. 49
  3.3.1 Standard Analysis Methods .................................... 49
  3.3.2 Related Work in Risk Assessment and Handling .......... 50
3.3.3 Discussion of Safety Approaches ........................................ 56
3.4 Hazards for Climbing Robots ........................................... 57
  3.4.1 Internal Hazards ..................................................... 59
  3.4.2 Collisions ............................................................ 60
  3.4.3 Driving into a Hole .................................................. 62
  3.4.4 General Robot Roll and Tilt ....................................... 63
  3.4.5 Wheel Abrasion and Wheel Slip ................................... 65
  3.4.6 Leaky Sealing ....................................................... 68
  3.4.7 Robot Tilt ........................................................... 68
  3.4.8 Robot Slip .......................................................... 70
  3.4.9 Discussion of Hazards .............................................. 71

4 Thesis .............................................................................. 73

5 Safety Analysis and Concept ............................................. 77
  5.1 Offline Safety Analysis .................................................. 78
    5.1.1 Fault Tree Analysis ................................................ 78
    5.1.2 Obvious and Ubiquitous Risks ..................................... 81
    5.1.3 High Dynamic and Hard Predictable Risks ....................... 82
  5.2 Safety Requirements ................................................... 87
    5.2.1 Design and Hardware Requirements ............................. 87
    5.2.2 Online Safety Requirements ...................................... 89
    5.2.3 Additional Requirements for Hard Problems ................... 91
  5.3 Safety Concept .......................................................... 92

6 Advanced Motion Control for Safe Navigation ....................... 95
  6.1 Ambiguity and Boundary Conditions ................................ 97
    6.1.1 Avoid Wheel Turning on the Spot ............................... 97
    6.1.2 Minimize Steering ................................................ 97
    6.1.3 Experimental Results and Advantages .......................... 99
  6.2 Traction Control System .............................................. 100
    6.2.1 Functionality and Integration ................................... 101
    6.2.2 Experimental Results and Benefit for Safety ................. 102
  6.3 Shear Force Controlling ............................................. 105
    6.3.1 Functionality and Integration ................................... 105
    6.3.2 Experimental Results and Benefit for Safety ................. 107
  6.4 Setup of the Advanced Motion Control Components .............. 110
7 Behavior-Based Adhesion Control .................................................. 113
  7.1 Behavioral Concept for Adhesion Control .................................. 113
    7.1.1 The Adhesion Control System ........................................... 114
    7.1.2 Behaviors and Modules .................................................... 115
  7.2 Chamber and Reservoir Control .............................................. 118
    7.2.1 Basic Chamber Control Behavior ...................................... 118
    7.2.2 Chamber Control Group .................................................... 121
    7.2.3 Reservoir Control .......................................................... 121
    7.2.4 Experimental Results and Benefit for Safety ....................... 122
  7.3 Chamber Deactivation .......................................................... 123
    7.3.1 Leakage Estimation Behavior ........................................... 124
    7.3.2 Pressure Trend Behavior ................................................ 125
    7.3.3 Basic Chamber Deactivation Behavior ................................ 127
    7.3.4 Chamber Deactivation Control ......................................... 130
    7.3.5 Experimental Results and Benefit for Safety ....................... 135
  7.4 Adhesion Force Control .......................................................... 139
    7.4.1 Force Value Control ....................................................... 140
    7.4.2 Force Point Control ....................................................... 141
    7.4.3 Force Control Group ....................................................... 142
    7.4.4 Calculation of Individual Chamber Pressures ....................... 142
    7.4.5 Experimental Results and Benefit for Safety ....................... 152
  7.5 Adhesion Network and Behavior Interaction ................................ 153
    7.5.1 Force Control ............................................................... 155
    7.5.2 Chamber Control ............................................................ 156
    7.5.3 Chamber Controllers and Deactivation Behaviors ................. 157
    7.5.4 Experimental Results ..................................................... 158
    7.5.5 Benefit for Robot Safety ................................................ 162

8 Online-Analysis of Safety ........................................................... 165
  8.1 General Concept ................................................................. 166
    8.1.1 Evaluation Function ....................................................... 167
    8.1.2 Risk Prediction ............................................................. 168
  8.2 Optimization Problem .......................................................... 169
    8.2.1 Degree of Current Adhesion ............................................. 170
    8.2.2 Training Data ............................................................... 171
    8.2.3 Learning Methods .......................................................... 172
8.3 Learning via Evolutionary Techniques ........................................... 174
  8.3.1 Introduction to Genetic Algorithms ........................................ 174
  8.3.2 Related Work in Genetic Approaches ..................................... 176
  8.3.3 Genetic Algorithm for Optimization ....................................... 178
  8.3.4 Rating of Individuals ....................................................... 179
  8.3.5 Fitness Function and Selection ............................................ 183
  8.3.6 Genetic Operations ........................................................... 184
  8.3.7 Elitism ............................................................................. 186
  8.3.8 Breakpoint of the Optimization Process ................................. 187
8.4 Experimental Results ................................................................. 187
  8.4.1 Experimental Setup ............................................................. 188
  8.4.2 Optimization Process .......................................................... 189
  8.4.3 Risk Prediction ................................................................. 193
  8.4.4 Limitations ........................................................................ 196
  8.4.5 Situational Risk Prediction .................................................. 197
  8.4.6 Discussion ...................................................................... 200

9 Risk Handling Measures and Strategies ........................................ 201
  9.1 Avoidance of Obvious Risks ..................................................... 201
    9.1.1 Detection of Obstacles .................................................... 202
    9.1.2 Behaviors for Evasive Actions .......................................... 206
    9.1.3 Experimental Results ...................................................... 208
  9.2 Counteractive Measures ............................................................ 211
    9.2.1 Emergency Stop ............................................................. 211
    9.2.2 Draw Back ..................................................................... 213
    9.2.3 Integration into the Motion Control Network ....................... 215
    9.2.4 An Obsolete Driving Strategy .......................................... 216
  9.3 Experimental Results of Risk Prediction .................................... 217
    9.3.1 Experimental Setup and Preparation .................................. 218
    9.3.2 Risk Prediction and Avoidance ......................................... 219
    9.3.3 Safety Measures on a Concrete Wall .................................. 222
  9.4 Discussion of Risk Handling Measures ....................................... 224

10 Conclusion .................................................................................. 227
  10.1 Summary ............................................................................. 227
  10.2 Discussion of Achievements ................................................... 230
  10.3 Future Perspectives ................................................................ 232
A Appendix 235

A.1 Mathematical Notation 235
  A.1.1 Special Notations 236
  A.1.2 General Mathematic Functions 237
  A.1.3 Statistics 237
  A.1.4 Matrix Operations 238
  A.1.5 General Evaluation Functions 239
A.2 Symbols 242
A.3 Key Data 243
A.4 Detailed Derivations 244
  A.4.1 Kinematic Constraints 244
  A.4.2 Calculation of Robot Roll and Tilt 245
  A.4.3 Direct Calculation of Chamber Forces 249
A.5 Safeguarding and Safety Analysis 251
  A.5.1 Standardized Safeguarding Process 251
  A.5.2 Overview of Standard Analysis Methods 252
A.6 Modular Control Architecture mca2-KL 256
  A.6.1 Software Components 256
  A.6.2 mcaBrowser 257
  A.6.3 mcaGui 258
A.7 Implementation Details 260
  A.7.1 Control Software 261
  A.7.2 Simulation 262
A.8 Behavior Values and Acquired Weights for Risk Prediction 268
  A.8.1 Weights Used in Simulation Environment 270
  A.8.2 Weights Applied on Real Prototype 271
A.9 Optimization Process 272
  A.9.1 Mutation Parameters 272
  A.9.2 Population Size 275
  A.9.3 Contradictive Training Sets 276

Bibliography 277

Index 291
1. Introduction

“Anything that can go wrong, will go wrong.”

Murphy’s Law \(^1\)

This quote, which has its origin in 1949, goes back on Edward A. Murphy Jr. who was an American engineer working on safety-critical systems in the range of rocketry and aerospace. Murphy’s Law became very famous in the range of error prevention, reliability, and further safety aspects of complex technical systems. The main idea and motivation behind this quote is that a developer should take care of all kinds of failures and error sources to achieve a certain amount of safety – particularly with regard to even impossible events, which would occur anyway in the worst case. In general, this challenge addresses all kinds of technical systems like e.g. industrial machines, software systems or networks of embedded systems. Related to Murphy’s Law, developers of technical systems have to spend much effort into safety aspects to break the law and to make not all errors happen. Nevertheless, “one hundred percent safety is impossible through technology”\(^2\), as Hirochika Inoue from the University of Tokyo pointed out. So the challenge is not to create a safe system, but one that is safe enough for the given task.

In contrast to the given examples – which are more or less closed systems – mobile robots have the great challenge to work in an open world which makes them very complex, but also highly safety-critical. On the one hand, they have to be developed in a way that they will not harm any persons or damage the environment, on the other hand, they also have to take care for their own safety. The main problem is based on the fact that robots interact with their environment whose influence on the robot can not be described well or, which in total behaves in a more or less unpredictable way. Therefore, not only internal malfunctions in hard- or software may occur, but also unconsidered situations, which have to be handled by the robot in an adequate way. One of these situations might be a patch of surface which is unknown for the robot since it has not been expected by the developer.

\(^1\)http://en.wikipedia.org/wiki/Murphy’s_law
as depicted in figure 1.1. Even if the system knows how to handle a similar situation, it is not said that this solution might also work in the present case, and whether the path could be passed by the robot or not.

Figure 1.1: Can the chosen path be handled by the robot? Which situations would cause the system to fail? Can these situations be described?

This thesis deals with safety aspects of wall climbing-robots, which have to ensure their own adhesion to a vertical structure. Here, a malfunction, an unhandled event, or a wrong evaluation of the situation can lead to a drop-off and therefore to a total loss of the system. This makes it inevitable to take a deeper look at possible error sources and to analyze the system and its components in depth. Due to the increased system complexity compared to ground vehicles – since not only the locomotion, but also the adhesion components interact with the environment – additional measures are needed to ensure, or at least improve navigation safety of the robot, as it will be discussed and presented in the scope of this thesis. As pointed out before, there exists no system which is one hundred percent failsafe and there will always be a factor of "just plain bad luck"\(^3\), as Evin Stump mentioned. The goal is to reduce the probability of a robot drop-off to a minimum by performing a detailed hazard analysis and applying special control components.

1.1 Problem Description

In the area of ground-based vehicles the description of events and their impact on the system is in the focus of researchers all over the world over many years. With increased computational power and better sensor technology robot developers were able to realize terrain classifications and adaptions of the navigation depending on an estimated or measured impact [Castelnovi2005, Stavens2006, Sun2005]. Based on geometric considerations, the influence of different obstacles or situations on a mobile robot can be described well. But, so far, it is nearly unexplored in which cases wall-climbing robots fail and in which way these cases can either be avoided, or at least reduced in their impact or probability of occurance. This thesis will examine possible events and errors, which could occur in the range of wall-climbing robots, and present suitable approaches to handle them. Of course, the whole area of climbing robots, as it will be introduced in section 2.3, would be too wide to handle, so the focus lies here on systems which are able to adhere to concrete

walls via negative pressure adhesion and use a drive system for navigation. Such systems can be used to perform different inspection or maintenance tasks at large concrete buildings like bridge pylons (figure 1.2), cooling towers or dams as they are requested by law periodically.

![Typical concrete bridge](image-url)

**Figure 1.2:** Typical concrete bridge (top), which has to be maintained and inspected periodically by the technical staff (bottom) \(^4\).

In contrast to other robots using legs or sliding frame mechanisms for locomotion, especially this combination of negative pressure adhesion and a drive, makes such systems highly critical with respect to its adhesion, since it cannot test adhesion points before passing them. It is also not possible to create an adhesion force as high as possible since this would lead to a stucked robot, which is no longer able to move. In fact the adhesion system has to balance inside of a small range between dropping off (downforce too low) and robot stuck (adhesion too strong). Furthermore, there are some more challenges in this area of research. At first, there is a lack of foresighted environmental sensors to measure the surface structure with a high resolution. Common existing sensors cannot be applied here since they either are too heavy, too large, or have a very small field of view. Even if those sensors could be used, the impact on different characteristics like roughness or step sizes on robot adhesion is unexplored, and leaves further space of research. Nevertheless, an indicator is needed whether the system will fail within the next seconds or not – either to start counteractive measures or to continue the current task execution. Beside permanent problems like e.g. wheel-slip, there exist hazards for robot adhesion, which are strongly related to surface characteristics and defects at the concrete building,

\(^4\)Images by courtesy of Borapa Ingenieurgesellschaft, Kaiserslautern, Germany
as they are depicted in figure 1.3. Sources of these hazards can be the general surface structure like exposed aggregate concrete, sheathing gaps, or edges, but also spalling or rock pockets, which are influenced by the weather and general stress. So far, it is not known in which way the negative pressure system is affected if the robot reaches these surfaces and whether it will fail or keep adhered. This is also caused by missing statistical data, which would be needed to create a model describing the interaction between robot and surface sufficiently.

![Images showing different surfaces and defects](image-url)

**Figure 1.3:** Different surfaces (a–e), construction characteristics (f–j) and defects (k–o) of concrete buildings [5].

### 1.2 Objectives

The scientific contribution of this thesis consists of the analysis of hazards for wall-climbing robots to identify potential sources of errors and ways to handle them. In fact, safe navigation of such a robot can only be achieved by methods of risk prediction to detect dangerous situations and by suitable measures to reduce or avoid these risks. To build up a climbing robot being able to navigate safely on vertical walls several issues have to be addressed. This includes a discussion of different hazards and their impact on the

---

5Images by courtesy of Borapa Ingenieurgesellschaft, Kaiserslautern, Germany
1.2. Objectives

robot – especially in terms of adhesion and navigation safety. Furthermore, several safety-related measures need to be described, which reduce the impact or the probability of occurrence of different hazards and improve navigation safety of the system. This includes general actions and adoptions, which are applied permanently during operation, as well as counteractive measures depending on the actual situation. Especially the developed combination of closed-loop control and behavior-based network is novel in the field of climbing robots. Here the most significant aspect is the online safety analysis based on information extracted from the adhesion control network, which allows a prediction of upcoming risks and to avoid situations, which would lead to a drop-off of the robot.

Although climbing robots are still a niche in the field of robotics, there exists a large number of different adhesion and locomotion principles. Therefore, it is not possible within the range of a single thesis to cover aspects of navigation safety for all these systems. This thesis will focus on aspects of wall-climbing robots using wheels for locomotion in combination with negative pressure adhesion. The following topics will be discussed in detail within this thesis and are important to develop a control system for safe navigation in general.

In-Depth Analysis of Hazards

As mentioned before, a climbing system using wheels in combination with negative pressure adhesion is highly critical with respect to its adhesion. In fact several possible error sources exist, which could lead to a total blackout or at least to severe damages at the robot. Although safety analysis of technical systems has a long lasting history and is well-known in industrial environments it is nearly unconsidered in the field of climbing robots. So far, researchers usually enhance already constructed systems in cases of new hazards without performing an in-depth analysis in the first place to get to know, which hazards may occur during operation. Within this thesis an analysis of hazards for wall-climbing robots is accomplished as a basis for suitable safety measures.

Safety Measures Related to Vertical Locomotion

Navigation in vertical environments needs to consider two aspects: Locomotion and adhesion. Both have a large influence on the operability and on the safety of the system. Based on the hazard analysis, suitable control methods need to be identified, which help to reduce either the probability of occurrence or the impact of a hazard on the robot system while it is in motion. Whereas some common approaches in the range of ground-based vehicles can be used as blueprint for suitable safety measures, others have to be developed from scratch. Research on locomotion control can help to improve motion capabilities, to increase system’s safety, and to reduce wear of high mortality components, as it will be handled in this thesis.

Developing of Components for Adhesion Safety

The adhesion system is the essential component ensuring safety of a wall-climbing robot. Although there exist several robots using negative pressure adhesion, only a handful of them apply approaches, which go beyond simple closed-loop control. For a semi-autonomous navigation, additional components need to be developed supporting the basic control structure via methods of force balancing and adhesion analysis. These components are elementary for robot adhesion, since simple closed-loop controllers are not able to cover all circumstances.
Online Safety Analysis and Suitable Counteractive Measures

It can be stated that – even with a more intelligent adhesion control – closed-loop controllers and redundant components might reduce the probability of certain hazards to occur but are still not sufficient to handle all of them. Therefore, a risk prediction approach needs to be developed, which allows a prior judgement of the system behavior depending on the surface to estimate upcoming problems. This online safety analysis is the basis to start suitable counteractive measures to avoid these hazards completely or to reduce their impact. Again, some reactions and strategies related to ground-based vehicles can be taken as a basis for some of these measures. As an example, a speed reduction in case of frontal obstacles can be applied to avoid or lower the impact of a collision.

Simulated Test Environment for Immediate Validation

Since driving on vertical concrete structures causes large stress on the prototypic climbing robot a suitable model has to be developed inside of a simulation environment. The simulation has to cover aspects like a thermodynamic model for simulating the airflow and to reflect the sealing's behavior to allow an offline development of safety and control algorithms. Of course, the simulation environment will not be one hundred percent realistic, but control components and safety measures can be tested without endangering the robot and used with adapted parameters also on a real prototype.

1.3 Outline

The next chapter 2 will present some fundamentals about the climbing robot CROMSCI, which is used as experimental platform, and some preliminary work. This includes the hardware structure and components, aspects of locomotion, kinematic calculations, as well as the already existing closed-loop controllers. Furthermore, this chapter introduces basic elements of the robot control software and gives an overview on the state-of-the-art of climbing robots.

Safety guidelines and hazards are the topic of chapter 3, which gives some general definitions and describes safeguarding processes in industry and robotics. Here, also some related work will be discussed, which deals with risk assessment and handling. The last section analyzes hazards for a wall-climbing robot like CROMSCI. This includes constant problems like wear of high-mortality parts, which occur permanently during operation, as well as situation based hazards. Afterwards, chapter 4 will postulate the key aspects of this thesis.

The offline safety analysis of the robot will be examined in chapter 5. Based on these results, some safety requirements will be derived, which should be considered and realized to achieve a certain system safety. Finally, the general concept for developing and implementing the safety components will be given. Afterwards, the omnipresent safety elements related to navigation and adhesion are discussed: Chapter 6 presents aspects and control components in the range of locomotion, which are helpful to optimize robot motion concerning velocity, transferable forces, power consumption, and wear of wheels. This is done by kinematic adaptions as well as novel traction and shear force controllers. The following chapter 7 introduces the new adhesion control network consisting of individual behaviors, which are responsible for certain tasks inside of the control structure.
This reaches from basic chamber control behaviors up to higher adhesion force control behaviors. Experimental results finish each individual section of both chapters to proof the operability of the developed elements.

Since the omnipresent safety measures are not sufficient in certain situations, chapter 8 will present the concept and realization of the risk prediction for online safety analysis. Here, the problem of predicting an upcoming drop-off will be discussed as well as different approaches to solve it. The chosen prediction function, which is acquired via an evolutionary algorithm, its optimization, and further adoptions are presented in detail. The last step for safe navigation are corresponding risk handling and avoidance strategies, which are triggered based on detected features or the developed risk prediction. Chapter 9 presents counteractive measures based on the online safety analysis of hard predicable hazards as well as a component to avoid obvious risks. Here, larger obstacles, which endanger the robot, are considered and their detection, analysis, and suitable reactions are introduced. Finally, experimental results of the real prototype CROMSCI according to the prediction of high dynamic risks and the counteractive measures are presented.

The final chapter 10 sums up the developed components and discusses the achieved results and scientific contribution of this thesis. Based on the reached goals also an outlook on future work will be given since there still exist a couple of possibilities for optimization. The appendix (chapter A) gives an overview on the used notation and symbols. It will also describe further details about the control software, the simulation environment, used parameters, and experimental results.
1. Introduction
2. Fundamentals

Climbing robots are in the focus of robotic researchers all over the world since the late eighties. These machines have been developed to fulfill different tasks in the range of maintenance, cleaning, inspection, or construction of buildings and other vertical structures like ship hulls or metal tanks. Nevertheless, only few climbing robots have been brought to application e.g. in ship industry or for window cleaning of high buildings. Especially climbing robots suitable to concrete structures are not commercially available yet, although they have a large field of application in terms of e.g. inspection tasks of buildings like dams, bridges, walls or cooling towers. The climbing robot CROMSCI should bring the development of these systems to the next level to disburden human technicians and to enhance building inspections in terms of economy, accuracy and safety.

Since this thesis is neither standalone in the wide range of robotics nor without any prerequisites, this chapter will introduce the climbing robot CROMSCI, which has been developed at the University of Kaiserslautern over the last years. This climbing robot is used as a demonstrator and test platform for experiments concerning climbing motions, to prove aspects and methods for safe navigation. The first section 2.1 will show its hardware components including the locomotion and adhesion mechanisms as well as closed-loop control elements in detail. This implies also the preliminary work by Carsten Hillenbrand who presented the mechatronic concept and the basic closed-loop control of the robot.

Section 2.2 will introduce the behavior-based control framework iB2C and shows how the robot’s motion control has been realized via a behavior network. Finally, section 2.3 gives an overview on different adhesion and locomotion principles, which can be found in the range of climbing robots, and shows according examples. The most suitable approaches will be discussed more detailed and compared to the present robot CROMSCI.

2.1 Climbing Robot CROMSCI

The wheel driven service robot CROMSCI has been build up to perform inspection tasks on large concrete buildings like bridge pylons, dams, or cooling towers area-wide and semi-autonomously. The system’s name stands for Climbing RObot with Multiple Sucking Chambers for Inspection tasks. So far, the system is able to climb on vertical walls via
closed-loop controlled negative pressure, as first introduced in terms of an early prototype in [Berns2004]. The application of such a robotic system implicates certain requirements, which have to be considered during development. Since they have a strong influence on the robot design, the most important key aspects and requirements are given as follows:

**Velocity and maneuverability** The vehicle speed and its ability to move are two main aspects in this field. Since CROMSCI should be able to move on large vertical structures, it needs a relatively high velocity even in vertical direction or overhead for a sufficient fast navigation between inspection points. Legged systems as presented by [Luk2001] are relatively slow and will not be able to go up a high river dam or bridge pylon in a short time. Another requirement is related to the used inspection sensors, since some of these systems (e.g. cover meter) need to be moved in a smooth and continuous way over the surface without rugged motions. This demand more or less limits the propulsion to a drive system.

**Payload** In literature (see section 2.3) some robotic systems exist, which are able to climb on concrete or similary uneven walls. Nevertheless, the given application has certain demands related to the payload in terms of inspection sensors like impact echo, cover meter or a Wenner probe. In contrast to simple cameras, which are very lightweighted and can be carried by nearly every kind of robot, these sensors have a weight of 10 kg and more. Therefore, the dimension of the robot as well as its adhesion and motion components need to be adapted according to the application.

**Usability** Velocity, maneuverability, and the capability of carrying a certain payload are important, but they are only the basis of the general operability of the system. To bring this robotic system into application it has to be more powerful, more efficient and less dangerous than common inspection approaches. This includes also aspects like maintainability and a broad range of handable tasks. Therefore, it must be able to carry different inspection sensors depending on the desired task, high mortality parts need to be easily replaceable, and the operation must be faster and less complicated compared to existing approaches. So far, inspections of these vertical structures are made by hand by using complex access devices like cranes or gondolas, as depicted in figure 1.2 and presented in [Berns2004], to reach the desired position for inspection. In some cases also professional climbers or large scaffolds are used to bring the technician to the building.

**Reliability and safety** Beside all functional aspects, the system has to ensure a certain robustness to be applicable\(^1\). These requirements include robust hardware, optimal controllers and methods to detect and handle hazardous situations and to recover from them. Finally, the system has to be secured via a cable or rope to eliminate the danger of a drop-off, which could harm persons and destroy the robot. But, the system itself should be safe enough to ensure its adhesion, since even a controlled drop-off might become dangerous.

To achieve the desired mobility and payload performance the robot CROMSCI combines several innovative technologies. For adhesion it uses an active negative pressure system

---

\(^1\)If it fails e.g. every five minutes it would not be usable in practice.
2.1. Climbing Robot CROMSCI

consisting of three suction engines, one large vacuum reservoir, and seven individual working chambers generating the downforce. An inflatable active sealing made of a rubber tube and a special coating to improve sliding characteristics is used to proof the chambers towards the ambient pressure. Three single steerable and driven wheels are used for high mobility in three degrees of freedom [Hillenbrand2008]. By turning the wheels, the whole robot glides over the concrete surface so that – in general – all seven vacuum chambers contribute to robot’s adhesion.

![Climbing robot CROMSCI attached to a concrete wall.](http://agrosy.cs.uni-kl.de/cromsci/)

The round chassis of CROMSCI (figure 2.1) is made of fiber glass and has a diameter of 80 cm. Heavy parts like wheels and suction engines are located inside of the lower part (yellow) of the robot to keep the center of mass close to the wall. On top lies the vacuum reservoir (red) including pressure sensors and valves for closed-loop pressure control and a movable manipulator arm, which carries the sensors for inspection. The overall weight of the system is at about 50 kg including the manipulator arm with an additional payload for inspection sensors of about 10 kg. CROMSCI has to be connected via an umbilical cord to a ground station because of high energy consumption of the suction engines and for communication purposes. The individual components will be introduced more detailed in the following sections. Further images and videos can be found on the robot’s website.

2.1.1 Components for Locomotion

To meet the requirements of high velocity and maneuverability CROMSCI is – in contrast to most other climbing robots using legs or sliding frame mechanisms for locomotion [Hong2009, Yong2006, Zhang2009] – equipped with an omnidirectional drive system. This section will introduce the special hardware components, which are used for locomotion as well as necessary calculations to determine the forward and backward kinematics and robot odometry [Schmidt2011].

Steerable Standard Wheels

The drive system of CROMSCI consists of three independently driven and unsprung steerable standard wheels [Hillenbrand2006]. Each wheel is located inside of a vacuum chamber and assembled of several components, as shown in figure 2.2: An outer cylinder is mounted

---

2http://agrosy.cs.uni-kl.de/cromsci/
fix to the robot chassis and guides the rotating inner wheel dome. The complete inner cylinder is driven by an integrated DC motor and is responsible for the wheel’s steering. All needed control circuit and amplifier boards are embedded into the inner wheel dome. The electronic connection to the robot case is realized via a collector ring linking outer and inner dome. A modern torque motor – responsible for movement – is mounted inside of the inner cylinder and connected to a Harmonic Drive gear with a reduction of 121:1. This combination of motor and gear is able to generate high movement forces per wheel (wheel radius $r_w = 0.053$ m, wheel width $d_{W/y} = 0.016$ m) at a maximum speed of $v_{w}^{max} = 0.1605$ m/s, which corresponds to 9.63 m/min. Due to the fact that the wheel is located inside of a vacuum chamber, the inner wheel dome has to be airproof to the outer one. In this early prototype the wheel is able to overcome obstacle of a maximum height of 6 mm because of a metal frame at the wheel mounting.

![Figure 2.2: Construction drawing of the propulsion unit.](image)

For safe adhesion, but also for controlling the robot’s traction, it is necessary to measure forces at the contact point between robot and wall. To get these information each wheel is equipped with an integrated load cell using strain gauges. These load cells are designed to detect vertical forces in z-direction up to $F_{w}^{max} = 2.000$ N and transverse torques around x/y-axis up to ±400 N with a high accuracy. The load cell itself is positioned 0.12 m above the driven wheel and connects it to the inner wheel dome. Therefore, all existing forces are going through the load cell and the robot is able to recognize whether downforces are too high for movement or dangerously low. The complete diameter of the wheel unit is 0.232 m and its height is 0.19 m. The total weight of the drive including all components is about 4.4 kg.

**Closed-Loop Drive Controllers**

To execute certain motion commands several closed-loop controllers have been implemented. The basic connection between control software and the motor hardware are DSP boards, which execute closed-loop PID control. They take the desired values and try to adjust the motor performance according to these values. Figure 2.3 shows the general basic motor controller of a single motor. In the current version, CROMSCI uses a cascaded controller for position and velocity. The position controller is of type P and
uses parameter $\kappa_sP$ as amplification factor. An integral portion is not needed due to the underlying PI velocity controller with parameters $\kappa_v dP$, $\kappa_v aP$ and $\kappa_v I$. In this case it is a so-called ReDuS controller\(^3\), which uses two proportional factors for a better performance [Weber2002].

\(^3\)Closed-loop control with adjustable damping and velocity (Regelung mit einstellbarer Dämpfung und Schnelligkeit)

---

**Figure 2.3:** General structure of a basic motor controller with cascaded controllers for velocity and position control, as introduced by [Hillenbrand2009].

In the available architecture, each DSP board is able to control two motors (as shown later in section 6.4 in figure 6.13) using two basic motor controllers. In the current setup, a basic P position controller for the wheel dome rotation is used as well as a basic PI velocity controller for wheel velocity. The power and encoder electronics include amplifiers, pulse-width modulation modules and other desired circuits for signal processing. A more detailed description of the single motor controller can be found in the thesis of Carsten Hillenbrand [Hillenbrand2009]. These position and velocity controllers provide some basic safety functionalities, since they are responsible for balancing the wheel speed and orientation. If they do not work properly the robot would either not drive the desired trajectory or, if e.g. one of them differs from the others, wheel slip would increase due to differences in wheel velocities or wrong steering angles.

**Omnidirectional Drive System**

Of course, the drive system has to perform a controlled robot motion based on individual steering angles and velocities of the single drive units according to the kinematic description. The model of the wheel configuration is illustrated in figure 2.4 with the kinematic center in the middle of the robot. The wheel $W_1$ at the front of the robot is located on the robot’s x-axis, the two rear wheels have a displacement to the front wheel of $\pm 120^\circ$. 

Here, $d_W$ is the distance of each wheel to the robot center, $\alpha_{W_i}$ describes the position of the wheel and $\beta_{W_i}$ the relative steering angle. According to figure 2.4 the position angles for CROMSCI are $\alpha_{W_1} = 0^\circ$, $\alpha_{W_2} = 120^\circ$ and $\alpha_{W_3} = -120^\circ$, the distance between wheel and robot center is $d_W = 0.26$ m for all wheels. All these parameters are needed to set up the kinematic equations based on motion constraints.

Figure 2.4: Wheel setup of the drive system.

### 2.1.2 Drive Kinematics

The used omnidirection drive system allows a higher mobility of the robot compared to other climbing systems using differential drive [Journee2011, Longo2006, Miyake2007]. Of course, this mobility comes with a more complex kinematic. The first step to set up the kinematic equations is to determine the kinematic conditions of standard wheels based on rolling and sliding constraints as given in [Siegwart2004]. Based on this, the equations for the forward (direct) and backward (inverse) kinematics can be created.

**Backward Kinematics**

Goal of the backward kinematic calculations is to determine single wheel velocities $v_{W_i}$ and wheel orientations $\varphi_{W_i}$ with $i \in \{1, 2, 3\}$ based on the desired linear velocities $v_{R|x}$ and $v_{R|y}$ and on the rotational velocity $\omega_R$ of the kinematic center of the robot. Corresponding to the general notation in appendix A.1 $v_{R|x}$ and $v_{R|y}$ denote the single components of the robot’s ($R$) velocity in x- and y-direction. The steering angles $\beta_{W_i}$ can be determined by applying the sliding constraints, as given in equation (2.1). These constraints point out that there will be no sideward motion transverse to the wheel’s rolling direction. More details about the rolling and sliding constraints can be found in appendix section A.4.1.

\[
\beta_{W_i} = \text{atan2}\left( \frac{\cos(\alpha_{W_i}) \cdot v_{R|x} + \sin(\alpha_{W_i}) \cdot v_{R|y}}{\sin(\alpha_{W_i}) \cdot v_{R|x} - \cos(\alpha_{W_i}) \cdot v_{R|y} - d_W \cdot \omega_R} \right) \tag{2.1}
\]
For better understanding absolute steering angles $\varphi_W$ are used, which describe the steering angle of wheel $W_i$ relative to the robot frame. If the robot simply drives forward all three angles are zero. This angle $\varphi_W$ can be determined for each wheel $i$ according to equation (2.2):

$$\varphi_W = \alpha_W + \beta_W - \frac{\pi}{2} \quad (2.2)$$

Now, the velocity $v_W$ of wheel $W_i$ in rolling direction can be determined on the basis of the rolling constraints (equation (A.20)). Equation (2.3) shows the calculation based on the desired robot velocity and the known angles $\beta_W$ from equation (2.1):

$$v_W = \sin(\alpha_W + \varphi_W) \cdot v_{Rx} - \cos(\alpha_W + \varphi_W) \cdot v_{Ry} - d_W \cdot \cos(\beta_W) \cdot \omega_R \quad (2.3)$$

Based on these equations, it is not possible to calculate the individual turning velocity and orientation of each steerable standard wheel depending on the desired robot velocity.

**Forward Kinematics**

Unfortunately the forward kinematic of CROMSCI cannot be determined by inverting the equations from the backward kinematics. The reason for this are intermediate wheel positions as shown in figure 2.5, which result in an invalid state in which the turning point of the robot is not unique. This happens e.g. during changes in the robot’s velocity and rotation. As shown in figure 2.5.b the elongations of the wheel axes do not necessarily meet within one single intersection point. In fact it lies somewhere inside of the triangle, which is described by the three red intersection points defined by all combinations of pairwise crossing. The resulting movement of the robot is not predictable by regarding the kinematic constraints. Therefore, the linear and angular velocities of the robot have to be calculated according to equation (2.4), in which the robot velocity $\vec{v}_R$ is the average of the current single wheel velocities $v_W$ transformed by steering angles $\varphi_W$.

![Figure 2.5](image-url)
For calculating the rotational velocity of the robot the tangential component \( v_{W_i|\text{tan}} \) of each wheel velocity (figure 2.6) is needed as given in equation (2.5). Each wheel \( i \) supports the robot rotation based on its kinematic parameters \( d_W \) and \( \alpha_{W_i} \), the current steering angle \( \varphi_{W_i} \) as given in equation (2.2), and the wheel velocity \( v_{W_i} \).

\[
v_{W_i|\text{tan}} = v_{W_i} \cdot \cos (\beta_{W_i}) = v_{W_i} \cdot \sin (\alpha_{W_i} - \varphi_{W_i})
\]

(2.5)

\[\begin{align*}
\vec{v}_R &= \begin{pmatrix} v_{R|x} \\ v_{R|y} \end{pmatrix} = \frac{1}{3} \cdot \sum_{i=1}^{3} \left( \begin{array}{c}
\cos(\varphi_{W_i}) \cdot v_{W_i} \\
\sin(\varphi_{W_i}) \cdot v_{W_i}
\end{array} \right) \\
\omega_R &= \frac{1}{3} \sum_{i=1}^{3} \left( \begin{array}{c}
-\frac{v_{W_i|\text{tan}}}{d_W} \\
\frac{v_{W_i} \cdot \sin (\varphi_{W_i} - \alpha_{W_i})}{3 \cdot d_W}
\end{array} \right)
\end{align*}\]

(2.4)

(2.6)

Figure 2.6: The wheel velocity \( v_{W_i} \) can be split up into a tangential \( v_{W_i|\text{tan}} \) and a radial component \( v_{W_i|\text{rad}} \).

The final rotational velocity \( \omega_R \) of the robot is again the average of all three tangential wheel portions of the wheel velocity as given in equation (2.6). It has to be noticed that a positive wheel velocity leads to a clockwise robot rotation if the steering angle \( \beta_{W_i} \) is zero. Therefore, the tangential wheel velocity \( v_{W_i|\text{tan}} \) is inverted:

Of course, the main problem of this calculation method results from the assumption that all wheels have the same traction. If one wheel is located on a slippy ground while the two others have grip the single wheel would only contribute a small portion to robot motion. Because of a general wheel slip, which is unknown, and the application scenario for CROMSCI it is valid to assume the same friction coefficient for all wheels. Finally, the results of the direct kinematics are used as basis for odometry calculations.
Odometry

Based on the calculations of the forward kinematics, the robot pose in the plane space relative to the starting point can be estimated. The process of dead reckoning is very common in navigation to get a first hint on the current position of the vehicle via summing up intermediate position and orientation changes. Since the previous calculations have taken place in the robot coordinate system, there now must be a distinction between the environmental $[E]$ and robot $[R]$ coordinates. The assumption that is made for odometry calculations in general is that the robot drives with constant linear and angular velocities during one calculation step. Then, equation (2.7) shows the determination of the actual robot orientation $[^E] \theta_R(t)$ in environmental coordinates at time step $t$. Obviously, this depends on the previous orientation $[^E] \theta_R(t-1)$, the duration of a calculation step $^4 \Delta t$, and the current rotational velocity $[^R] \omega_R(t)$ in robot coordinates, as calculated in previous equation (2.6). Additionally, $[^E] \theta_R^{\text{init}}$ denotes the initial orientation of the robot at the beginning of the localization procedure.

\[
[^E] \theta_R(t) = [^E] \theta_R(t-1) + \Delta t \cdot [^R] \omega_R(t) \tag{2.7}
\]

\[
[^E] \theta_R(0) = [^E] \theta_R^{\text{init}}
\]

The calculation of the robot’s position $[^E] \vec{P}_R(t)$ have to take into account that CROMSCI has an omnidirectional drive and does not move only in x-direction like vehicles with Ackermann steering or differential drive. Figure 2.7 depicts the geometric relationships between the previous (red) and current (green) vehicle position. As mentioned before it is assumed that the robot drives on the dashed arc within one calculation step. The position $[^E] \vec{P}_R(t)$ at current timestep $t$ can be calculated according to equation (2.8) with the previous position $[^E] \vec{P}_R(t-1)$, the rotation matrix $R$ using the previous robot orientation $[^E] \theta_R(t-1)$, and the current displacements $[^R] \Delta x_R(t)$ and $[^R] \Delta y_R(t)$ in x-direction of the robot and y-direction, respectively. Again, $[^E] x_R^{\text{init}}$ and $[^E] y_R^{\text{init}}$ denote initial position values on startup.

\[
[^E] \vec{P}_R(t) = [^E] \vec{P}_R(t-1) + R([^E] \theta_R(t-1)) \cdot \begin{pmatrix} [^R] \Delta x_R(t) \\ [^R] \Delta y_R(t) \end{pmatrix} \tag{2.8}
\]

\[
[^E] \vec{P}_R(0) = \begin{pmatrix} [^E] x_R^{\text{init}} \\ [^E] y_R^{\text{init}} \end{pmatrix}
\]

The calculation of the robot-related position change depends on the rotational velocity of the system. If the robot just drives in straight direction without turning ($[^R] \omega_R(t) = 0$), then it is just the multiplication of the linear velocities with the elapsed time $\Delta t$ as given in the upper case in equation (2.9). Otherwise, the length of the displacement $h_t$ has to be calculated and split up into x- and y-components in robot coordinates. According to figure 2.7 $[^E] \vec{P}_t$ denotes the turning point, $r_t$ is the radius of the arc the robot drives on, $\alpha_t$ is the half angle of motion and $\alpha_v$ is the direction of the velocity vector $[^R] \vec{v}_R(t)$.

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]
\[
\begin{pmatrix}
[R] \Delta x_R(t) \\
[R] \Delta y_R(t)
\end{pmatrix} = \begin{cases}
\Delta t \cdot \begin{pmatrix}
[R] v_{Rx}(t) \\
[R] v_{Ry}(t)
\end{pmatrix}, & \text{if } |[R] \omega_R(t)| = 0 \\
2 \cdot h_t \cdot \begin{pmatrix}
\cos (\alpha_t + \alpha_v) \\
\sin (\alpha_t + \alpha_v)
\end{pmatrix}, & \text{else}
\end{cases}
\] (2.9)

with
\[
\alpha_t = \Delta t \cdot \frac{|[R] \omega_R(t)|}{2} \\
\alpha_v = \arctan \left( \frac{|[R] v_{Ry}(t)|}{|[R] v_{Rx}(t)|} \right) \\
h_t = \sin(\alpha_t) \cdot r_t \\
r_t = \frac{\sqrt{[R] v_{Rx}^2(t) + [R] v_{Ry}^2(t)}}{|[R] \omega_R(t)|}
\]

The resulting displacement is a first hint on the robot position while it is operating. Nevertheless, dead reckoning comes with the big disadvantage, that the calculated pose differs from the real one more and more with increasing runtime. This is a general problem due to small errors during the single calculation steps, which are cumulative and sum up to a large mismatch after some time. Therefore, odometry values have to be supported by absolute localization methods such as GPS or active beacons. So far odometry is used only for local mapping as it will be presented later on in section 9.1.

### 2.1.3 Adhesion System

As introduced before the climbing robot CROMSCI makes use of negative pressure for the adhesion to vertical or overhead structures. The main challenge is here that the robot should neither stick to the wall if the adhesion forces are strong nor fall down if they...
are too low. The adhesion system itself consists of seven single vacuum chambers, which glide over the surface during robot motion, as depicted in figure 2.8. They are supported by one large reservoir chamber on top with a volume of about 20 l, which is evacuated by three strong self-cooling suction engines (3600 W in total). Inside of the chambers $C_1$, $C_3$ and $C_5$ the three omnidirectional wheels with integrated load cells are mounted, but do not influence the negative pressure system. Each of the seven working chambers with a volume of about 4.5 l receives its negative pressure from this reservoir. This amount of working chambers is a compromise between realization and control simplicity, which demands the use of only one large chamber, in contrast to operation safety, which needs as many chambers as possible. Some considerations about this reason balancing can be found in the work of Hillenbrand as well as some more detailed descriptions about the system components and closed-loop controllers [Hillenbrand2009]. Nevertheless, this section will shortly introduce the most important and innovative hardware components, which are necessary for robot adhesion as well as the basic control elements as they have been presented by Hillenbrand, which form the starting point of the adhesion control system introduced in this thesis.

![Figure 2.8: Bottom view on CROMSCI showing the seven adhesion chambers, wheel positions and the sealing in shape of spokes.](image)

**Negative Pressure System**

The mechanical structure of the negative pressure system is a complex entity of glass fiber volumes, tubes, valves and further elements. Figure 2.9 shows a simplified model of the system with its seven working chambers marked with numbers 1 to 7 and the reservoir volume (R) on top. Here, an exemplary crack is shown, which influences chambers $C_3$ and $C_7$ by an increased leakage what results in a higher massflow $\dot{m}_{L_3}$ between outside and chamber $C_3$ and massflow $\dot{m}_{L_7}$ between the chambers itself. The remaining chambers are influenced only by some basic leakages, which are comparable low. To allow a closed-loop control of the chamber pressures, the adhesion chambers and the reservoir are equipped with pressure sensors and connected via valves. This allows an independent pressure control of each chamber via an adjustment of the valve areas $A_V$ with $i \in \{1, 2, \ldots, 7, R\}$, which changes the corresponding massflows $\dot{m}_V$. Of course, there exists a connection
between all working chambers caused by the sealings and its basic leakage areas $A_{Lij}$. Therefore, neighboring chambers will be also evacuated, if one chamber is evacuated, but the amount of this air flow does not have a strong effect on the system behavior. The closed-loop controller opens and closes the valves to evacuate the chambers independently on the basis of current leak tightness and the desired pressure. This control structure with reservoir volume and valves has been chosen because of too slow power adjustments of the suction engines, which generate a constant massflow $\dot{m}_A$. In cases of emergency these adhesion motors cannot increase pressure force fast enough to prevent the robot from falling down.

![Diagram](image)

**Figure 2.9:** Setup of suction area with an example crack below chambers $C_3$ and $C_7$ (top view) and a model of the negative pressure system and the airflows across leakages and valves (side view).

The total effective suction area of CROMSCI is about 0.4 m$^2$ while the needed pressure differences of the chambers lie between $-5000$ Pa and $-10000$ Pa compared to ambient pressure (100 000 Pa). Even with this low negative pressure of only 90%–95% compared to ambient air the resulting downforce lies in a range of approximately 2 000 N to 4 000 N. In experiments the robot actually could achieve downforces of up to 6 000 N. Of course, the robot will not be able to move if it is adhered that strongly, but this gives the robot sufficient reserves for situations in which one or more working chambers are loosing negative pressure and must be isolated from the vacuum system. By closing the valves of untight chambers a propagation of ambient pressure to the other chambers and the reservoir and therefore a total loss of adhesion can be avoided. Inactive chambers are tested from time to time if they still have high leakages or if they can be reintegrated into the negative pressure system [Hillenbrand2008].
The exact geometrical chamber setup is described in figure 2.10. Here, all outer downforce points – which are located in the geometric center of each chamber – lie on a circle and are spread symmetrically. Each chamber middle point \( \vec{P}_{C_i} \) can be calculated as shown in equation (2.10) by using the angle \( \alpha_{C_i} \) of the chamber and the distance \( d_{C_i} = 0.254 \text{ m} \) from the kinematic center, which is equal for all outer chambers \( i \in \{1, 2, 3, 4, 5, 6\} \). The angle \( \alpha_{C_i} \) varies over multiples of 60°. The center chamber \( C_7 \) is located exactly inside of the robot center so its distance \( d_{C_7} \) is zero. This description of chamber position is similar to the kinematic setup shown in section 2.1.1.

\[
\vec{P}_{C_i} = \begin{pmatrix} x_{C_i} \\ y_{C_i} \end{pmatrix} = \begin{pmatrix} \cos(\alpha_{C_i}) \cdot d_{C_i} \\ \sin(\alpha_{C_i}) \cdot d_{C_i} \end{pmatrix}
\] (2.10)

With the given chamber angles \( \alpha_{C_1} = 0^\circ, \alpha_{C_2} = 60^\circ, \alpha_{C_3} = 120^\circ, \alpha_{C_4} = 180^\circ, \alpha_{C_5} = 240^\circ, \alpha_{C_6} = 0^\circ \) and \( \alpha_{C_7} = 0^\circ \), the symmetrical robot setup leads to a downforce position at the robot center, if the negative pressures inside of all chambers are equal. This aspect also counts for groups of three symmetric chambers as illustrated as colored areas in figure 2.10 in terms of chambers \( C_1, C_3 \) and \( C_5 \) – as well as for chambers \( C_2, C_4 \) and \( C_6 \). This aspect becomes important for some calculations described in chapter 7.

Adhesion Control System

Up to this state of development, the adhesion system of CROMSCI uses classic closed-loop controllers. Figure 2.11 sums up the control elements as described in [Hillenbrand2009]. Nevertheless, it is necessary to introduce the already existing components for adhesion control to point out, which adaptations have been made as they will be presented in chapter 7. In general, two controllers are used, which are supported by different calculation
and estimation elements. The outer controller (marked as ‘out’ in figure 2.11) is responsible for balancing out robot tilt. To achieve this a robot tilt controller is used, which gives desired values of downforce \(F_N|_z\) and point of downforce \(\vec{P}_{FN}\) to the negative pressure system. These values depend on the given values from the user and the current values from the drive system \(F_{D|_z}\) and \(\vec{P}_{DN}\), which are computed by the force point calculation based on the wheel downforces \(F_{W_1...3}|_z\) measured by the load cells, which are depicted in the bottom left icon. A list of all symbols can be found in the appendix in table A.3.

![Diagram](image_url)

**Figure 2.11:** Existing components of the adhesion control system including closed-loop controllers to adjust robot tilt and chamber pressures and additional calculation modules.

The inner control structure (marked as ‘in’ in figure 2.11) tries to regulate the chamber pressures via the corresponding chamber pressure controller. This controller uses the values from the pressure sensors \(p_{C_1...8}\) for balancing and sets the valve areas. The current valve areas \(A_{V_1...8}\) in combination with the current pressure values are used by the leakage estimation to compute chamber leakages \(\hat{A}_{L_{1...7}}\). The chamber deactivation takes these leakage values to deactivate chambers in cases of high leakages. This is done by setting the minimum chamber pressure \(p_{C_{min}}\) of the affected chamber to the ambient pressure \(p_{amb}\) as given in equation (2.11). Here, \(p_{C_{min}}\) and \(A_L^{max}\) denote the theoretical minimum chamber pressure respectively the maximum threshold for chamber leakage.

\[
p_{C_{min}} = \begin{cases} 
p_{amb}, & \text{if } \hat{A}_{L_i} > A_L^{max} \\
p_{C_{min}}, & \text{else}
\end{cases}
\]  

(2.11)
The principle deactivation of chambers has been introduced first by Wettach who only simulated the system behavior [Wettach2004]. His work has been the foundation of the deactivator by Hillenbrand who successfully brought this approach into application on the real machine [Hillenbrand2009]. But, their concept of chamber control and chamber deactivation had some disadvantages related to safety analysis, extendability, and needs to be improved for higher robustness, quick reaction times and shorter inactive periods of the chambers. Wettach also introduced an iterative calculation method for the determination of the desired chamber pressures based on desired force values. This element was realized as *chamber pressure calculation* component, but has also been replaced by a new approach, as described in section 7.4.4, to be faster and more precise.

**Sealing System**

As already mentioned, CROMSCI uses an omnidirectional drive for locomotion. This implies that the vacuum chambers are moved over the surface without lifting them during locomotion as in the case of legged robots. Therefore, a flexible, but robust sealing mechanism, has been realized trying to optimize the best of two worlds: For good adhesion a very soft and flexible sealing would be best, which guarantees that the chambers are very tight. On the other hand hard and robust sealings are needed for good sliding characteristics, low wear and small chance of penetration. To achieve the best compromise between both worlds specials sealings have to be developed, which are wear-resistant, leak-proof and easy sliding, but allow controlled basic leakages at the sealing areas. In the current version, CROMSCI uses an air filled tube in the shape of spokes (figure 2.8), which produces a constant force to the sliding area independent to the ground shape. It is possible to vary the height of the sealing by changing the air pressure inside. During operation the sealing is used with a closed-loop controlled pressure difference of about 120 000 Pa (1.2 Bar) above the ambient pressure. This PI-controller has been implemented on a DSP circuit board and is able to adjust the desired sealing pressure very fast and accurate. The pressure again is a compromise, since the sealing becomes more leak-tight if it is pressed stronger against the surface, on the other hand it absorbs more adhesion forces, which results in higher friction during motion. This becomes obvious in detail in figure 2.12, which shows the inflated rubber, which seals the robot chassis to the vertical wall. Onto the surface of the sealing rubber a synthetic fiber material has been applied as sliding coat reducing friction and wear. The lift/drag ration and the value of friction have been examined on a Pin-on-Disc test stand [Hillenbrand2007, Leichner2008].

![Figure 2.12: Detailed view on the contact between surface and sealing.](image-url)
2.1.4 Inspection and Manipulation System

Since the robot is designed for inspection tasks mobility is a main requirement. This includes the drive system so that the robot is able to reach the desired locations at the building, but also a manipulation device to position the inspection sensors even in corners or other unapproachable areas. Therefore, a flexible manipulation device plays a tremendous role to fulfill inspection tasks effectively. For the inspection itself different sensors can be used like cover meters\(^5\), which are the most common method to detect rebars in the concrete and prescribed by law (e.g. in DIN 1045). Other sensors are impulse radar\(^6\) for detecting holes lying up to 2 m inside of the concrete, nuclear magnetic resonance (NMR) sensors\(^7\), or high-resolution cameras to spot cracks of 0.2 mm width. An overview of different sensor technologies, which can be used to inspect concrete walls can be found in [Weise2001].

![Diagram of robot with installed manipulator arm](image)

**Figure 2.13:** CROMSCI with installed manipulator arm on the circular mounting and integrated dummy sensor at the tool center point (TCP).

To carry and utilize the inspection sensors they are connected to a movable manipulator device, which is able to extend the robot range. This is necessary to inspect corners or edges of the building the robot cannot reach by driving and also enlarge the distance of inspection sensors to the robot’s engines, which may influence the measurements. The best and most compact solution for CROMSCI are round guides with two driven sledges, as shown in figure 2.13. By this, motions in x-y-plane are realized with a parallel kinematic. Therefore, one arm is mounted on each sledge with passive bearings and both arms are merged together at a conjunction point. Here additional motors allow vertical motion and a rotation of the sensor head below. This solution has the advantage that the middle of the robot is still usable for components like vacuum chambers and electronics. The disadvantages are the non-rectangular working space and correlations between motion elements of the arm, which leads to a more extensive motion control software [Hillenbrand2006b].

---

\(^5\)http://en.wikipedia.org/wiki/Cover_meter  
\(^6\)http://www.impulsegeophysics.com/ImpulseRadar.html  
\(^7\)http://www.magritek.com/applications-portable
2.1.5 Environmental Sensors

So far, the robot CROMSCI is not equipped with external sensors for delivering information on the environment. Only internal sensors like the strain gauges inside of the load cells, pressure sensors for adhesion system and scaling, as well as motor encoders have been considered. In the scope of this thesis, also this aspect has been handled and a detection of obstacles and corresponding reactions by the system have been realized, as presented later on in section 9.1. As environmental sensor a laser range sensor of type URG-04LX from Hokuyo Automation Co. has been selected, as depicted in figure 2.14. The key data of this sensor are an angle range of 240°, a measuring distance from 0.02 m to approximately 4.0 m, a scan time of 10 Hz and a distance accuracy of ±0.01 m in the used range of up to 1 m. The main advantages compared to other laser range sensors e.g. by Sick AG are a low weight of only 141 g and small dimensions of only 0.05 m × 0.05 m × 0.07 m. More information about the characteristics of this sensor can be found in the publications by the ETH Zurich [Kneip2009] and the University of Michigan [Okubo2009].

Figure 2.14: The applied Hokuyo URG-04LX laser range sensor [9].

Compared to the surface structure the accuracy is relatively low, since even small structures or a certain roughness might endanger robot adhesion. Therefore, the laser range sensor has been used only for the detection of large protrude obstacles or deep holes, as it will be presented in section 9.1. Of course, other more precise sensors have been evaluated in terms of the needed accuracy and applicability to detect small surface disturbances smaller than one millimeter. Most of the industrial devices with the required accuracy are much too heavy or too large and therefore cannot be applied here. In general, nearly all these systems use structured light in terms of e.g. a laser line or a grid projection in combination with cameras or other optical sensors.

Interesting devices, which can be used for surface analysis, are light-section sensors from Dr. D. Wehrhahn. Figure 2.15a shows the OPTimesse 2D-300 sensor, which measures the height of points on a laser line. Although the vertical accuracy with 0.133 mm is high enough, the device is not practical for the given application, since the maximal length of laser line and therefore the horizontal scan range is only 0.185 m, which is less than

---

9http://www.sick.com/
9http://www.hokuyo-aut.jp/
10http://www.drwehrhahn.de/
11http://www.gfm3d.com/
a fourth of the robot’s diameter. Since the sensor chassis has a dimension of about 0.05 m × 0.09 m × 0.30 m it is not possible to mount four or more of these sensors on the robot to achieve a capable measuring area.

Sensors developed by GFMesstechnik project structured light on the surface to get depth information with a high accuracy. Unfortunately these sensors have either a very small measuring area of only 0.04 m × 0.03 m as in the case of MikroCAD lite (figure 2.15b) or they are too heavy and too large (ShapeSCAN 3D in figure 2.15c). It becomes obvious, that no all-in-one device exists, which fulfills all desired aspects of measuring range, accuracy, update rate and mounting requirements. Based on all these reasons and the high price of more than 10000 € per surface sensor, the decision was to omit this kind of detection device and to use other internal mechanisms to identify critical situations, as they will be presented later.

2.2 Robot Control Software

For first experiments with CROMSCI a basic control software on the basis of the MCA2-KL framework, as presented in section A.6, was build up to steer the robot by hand and to set desired adhesion pressures of the seven working chambers. As it will be described later, several additional software elements were developed, which perform automatic operations, debugging and safety-related tasks. As basis for robot navigation a behavior-based network has been developed, which is responsible for the motion of the robot itself, but also for its manipulator arm. Below this control network the kinematic calculations as well as simple odometry localization is positioned according to the drive kinematics presented in section 2.1.2. The resulting steering and driving commands are given to the motor controllers embedded into the DSP units according to section 2.1.1, which perform the closed-loop motor control. Figure 2.16 gives an overview on the already existing (yellow) or actual developed components, which are needed for basic robot control.

On the right side of the control structure in figure 2.16 the adhesion control elements, which get the desired pressure values and perform a closed-loop control as presented in section 2.1.3, are located. This also includes the basic version of the deactivation of chambers in cases of high leakages and the chamber pressure distribution component by Wettach. Again, these components only allow a basic functionality of the robot, which is not very reliable.
2.2. Robot Control Software

2.2.1 Behavior-based Control Architecture iB2C

For behavior-based control of CROMSCI the iB2C architecture\footnote{http://rrlib.cs.uni-kl.de/mca2-kl/libraries/ib2c/} – which stands for Integrated Behavior-Based Control introduced by \cite{Proetzsch2010} – is used. This architecture has been developed at the University of Kaiserslautern over the last years and is applied to various complex robotic systems. Certain guidelines lead the developer through the process of design and realization.

Behavior Module

The basic unit of this architecture is the behavior module as shown in figure 2.17a. Each behavior module $B$ is an algorithmic element, which generates outputs from given inputs and can be described as a triple of an activity function $f_a$, a target rating function $f_r$ and a transfer function $F$ as given in equation (2.12):

$$B = (f_a, f_r, F) \quad (2.12)$$

In general, each behavior computes control data $\vec{u} \in \mathbb{R}^n$ via the transfer function $F(\vec{e}, \iota) = \vec{u}$, which uses the given input vector $\vec{e} \in \mathbb{R}^m$ and the activation $\iota \in [0, 1]$ of the behavior. Inputs can be sensor values as well as control data from other components. The activation value is one of the following set of meta data. These values are used internally to influence other behaviors or to signal the behavior’s state.
Figure 2.17: Components of the iB2C architecture: The fusion behavior (b) is derived from the basic behavior (a), which is the fundamental unit [Proetzsch2010].

Stimulation The stimulation $s \in [0, 1]$ is an input of a behavior, which determines its desired relevance. The value ranges from $s = 0$ (no stimulation) until $s = 1$ (full stimulation). In colloquial speech the stimulation allows the behavior to be active.

Inhibition Beside stimulating a behavior, it is also possible to reduce its influence. The inhibition $i = \max_{j=0,..,k-1}(t_j) \in [0, 1]$ is calculated based on the inhibiting input vector $\vec{r} \in [0, 1]^k$. In this case $i = 0$ means no inhibition while $i = 1$ represents a full inhibition.

Activation The activation $\iota \in [0, 1]$ is the effective relevance of the behavior and used only internally. It is the combination of stimulation and inhibition: $\iota = s \cdot (1 - i)$.

Activity The activity $a \in [0, 1]$ is the amount of action the behavior is performing. In the case of $a = 0$, the behavior is inactive while it has highest influence at the other end of the range ($a = 1$). The output vector $\vec{a} = (a, a_0, ...a_{l-2})^T \in [0, 1]^l$ allows the use of derived activity values $a_0, ...a_{l-2}$ with $a_j \leq a$, $\forall j \in \{0, ..., l-2\}$ to transfer only a part of the behavior activity to other behaviors and is calculated by the activity function $f_a(\vec{e}, \iota) = \vec{a}$

Target rating The target rating output $r \in [0, 1]$ indicates the satisfaction of the behavior with the current situation and is calculated by the target rating function $f_r(\vec{e}) = r$. Here $r = 0$ refers to a high contentment whereas $r = 1$ indicates a high dissatisfaction of the behavior.

This architecture also provides a set of design principles, which have to be taken into account. For example, behavior activity is limited by its activation ($a \leq \iota$) whereas the activation does not have any direct influence on target rating. Based on this, it is possible to implement a very complex system behavior via many individual and interacting units. For clearness behaviors can be encapsulated to behavior groups.

Fusion of Behaviors

If two or more behaviors try to access the same resource an arbitration mechanism is needed to decide, which values should be provided to the resource. For these cases fusion behaviors can be used allowing it to fuse output vectors of different behaviors based on
their meta data. Figure 2.17b shows such an element, which is a special behavior-based module. The calculation functions are predefined, only the type of fusion can be chosen: Maximum fusion, weighted fusion or weighted sum fusion. Figure 2.18 gives an example of data fusion where the outputs of three behaviors are combined.

![Diagram of data fusion](image1)

**Figure 2.18:** Exemplary fusion of the outputs of three behaviors by taking meta data into account [Proetzsch2010].

### Grouping of Behaviors

Another element for simplifying the network structure and for a better overview are behavioral groups. A behavior group is a collection of behaviors representing a semantic group. It acts like a normal behavior providing the same meta data as shown above. As depicted in figure 2.19 a group may contain a fusion behavior collecting data from multiple behaviors, which try to influence the same resource. In this case the meta data of this fusion behavior are connected to the outputs of the behavioral group. Another possibility is to use behavior outputs, which characterize the state of the whole group.

![Diagram of behavior group](image2)

**Figure 2.19:** Example for a behavior group, which contains two basic behaviors and a fusion behavior for arbitration and providing its meta data to the group [Proetzsch2010].

A more detailed description of iB2C and its components can be found in the thesis of Proetzsch [Proetzsch2010a]. He also introduced a couple of design principles, which are helpful in the development phase of a behavioral network.
2.2.2 Behavior Network for Motion Control

The motion control elements, which transfer user commands or automatic trajectories to robot velocities have been realized as behavior-based network using the iB2C components introduced before. Figure 2.20 depicts the control elements of the drive system. The more complex behaviors, which allow robot control via a (GUI-) joystick or perform random motions, are located on the top. The steering commands for all robot motions with three degrees of freedom (DOF) are given to a drive fusion module, which calculates one set of stimulation values for each of the six motions turn left (TL), turn right (TR), drive forward (F), drive backward (B), drive sideward left (L) and drive sideward right (R). A detailed view of the behaviors for straight and sideward motion is given in figure 2.21. A diagonal motion of the robot can then be realized by an overlay of straight and sideward motion.

![Figure 2.20: Overview on the implemented behavior network for basic motion control of the robot (the behaviors drive backward and drive left are not displayed here).](image-url)

Since the structure has to be extendable – additional behaviors should be able to influence these basic motions – a fusion layer (blue modules in the third line) has been added. By this mechanism further inputs can be connected to trigger each motion individually. The motion behaviors are stimulated by the activity of the corresponding fusion behavior. A fully activated motion behavior is associated to a full speed execution of the desired command. Since there always exist two behaviors, which are conflictive like turn left and turn right, an additional weighted fusion element is needed, which generates the desired velocity in a range of $[-1, 1]$ pointing out the final amount of turning or driving. It is important to know that the activity values of the fusion nodes as well as of the motion behaviors only represent the absolute turning or motion velocity relative to a maximum velocity in a range of $[0, 1]$ to be conform to the iB2C rules. The signed values are encoded...
in the general control output. As an example, the control output of *turn left* is 1 if its activity is also 1. In contrast, the control output of the opposed *turn right* behavior is $-1$ if the activity is 1. It should be obvious that there will not be any turning if both behaviors are active in the same manner.

![Diagram of basic behaviors for driving](image)

**Figure 2.21:** Detailed view on the behaviors responsible for driving.

A similar network has been realized for the control of the TCP, which has four DOF. Therefore, eight basic behaviors exist, which are corresponding for the motion of the manipulator arm: *turn head left, turn head right, move head up, move head down move head away, move head closer, move head left* and *move head right*. These commands are handled in the same way with higher control and fusion modules as well as final fusion modules for arbitration of opposed behaviors. Only the fusion layer above the basic behaviors is not implemented here, since the manipulator arm should not be moved by additional evasive behaviors. Because of the modular structure of the network a backfitting of these components is very simple if it becomes needed.

Afterwards, all motion values (given as behavior outputs $u$) lying in a range of $[-1, 1]$ have to be adapted according to the velocity limits of the individual motions. In this case it has to be considered that the motions depend on each other. If the manipulator arm e.g. is moved away from the robot center on full speed and an additional sideward motion is induced, both motions will be reduced in their velocity, since the sledges are at their limits. The same counts for the wheels of the robot whose turning speeds are a summation of linear and rotational velocities. Finally, the speed values are either given to the motor controllers as in the case of the TCP motions or used as inputs for the kinematic calculations of the omnidirectional drive.

## 2.3 Related Work in Wall-Climbing Robots

Of course, researchers all over the world are working on wall-climbing robots. The capabilities of these systems strongly depend on their tasks and on the applied adhesion principles and locomotion types. The next section will give a rough overview on climbing robots in general, sorted by their adhesion mechanism. Afterwards some interesting climbing systems using negative pressure adhesion in the area of concrete buildings will be described more detailed.

---

In literature a large number of robots is postulated as *climbing* robots, although they are only able to 'climb' a stair or a single step. Therefore, the term *wall-climbing* is used here for distinction.
2.3.1 Adhesion Principles and Locomotion

A common technique for climbing robots is magnetic adhesion. This includes electromagnets as well as permanent magnets, which are either positioned on the surface or kept with a certain distance away from it. Common applications of these systems are inspections, maintenance and construction work e.g. of high power poles, steel tanks or ship hulls. The principle is very reliable on ferromagnetic surfaces and it is able to create strong adhesion forces on a very small area. By using rare-earth magnets it is also possible to create adhesion without energy consumption. In this case the magnets either have to be peeled off or they have to stay within a certain distance from the wall to limit the magnetic forces and the friction. The locomotion principles used in combination with magnetic adhesion are e.g. legs, sliding frame mechanism and wheels. An often cited robot is REST, which is equipped with six legs [Grieco1998] and which has been developed for welding tasks (figure 2.22a).

Figure 2.22: Climbing robots with magnetic adhesion using legs (a–d), wheels with separate magnets (e,f), magnetic wheels (g–i) and sliding frame mechanism (j) for locomotion.

\[\text{14http://www.robotic.diees.unict.it/robots/robots.htm}\]
2.3. Related Work in Wall-Climbing Robots

Other legged systems are the rvc (reconfigurable vertical climber) using four legs with permanent magnets on a peeling pad [Peters2010], robinspec by [Fortuna1996], winspecbot [Kamagaluh2012] or the climbing machine by [Berengueres2007]. Another common locomotion type are wheels applied in addition to separate magnets as introduced e.g. by [Berns2004a] in terms of climbing robots for education or the so-called ndt robot by [Shang2008]. Another option is to combine magnets into the locomotion system and to use either magnetic tracks or wheels [Fischer2011, Krause2000]. Finally, also sliding frame mechanisms can often be found in combination with magnets e.g. for robots, which are designed to climb and inspect electrical towers [Golightly2002]. All the mentioned systems are displayed in figure 2.22. Of course, this adhesion principle is not applicable to concrete surface, but for ferromagnetic structures it is the best solution regarding energy efficiency, adhesive power and reliability. Especially walking robots equipped with permanent magnets are very safe, since they can test each foothold for adhesion and the magnetic forces are independent from energy supply.

The second technique – probably the most used adhesion method in the field of climbing robots – is pneumatic or negative pressure adhesion. Here it can be distinguished between three different types: Passive suction cups, active suction chambers and a vortex system. The last one is a patented adhesion method known as vram (vortex regenerative air movement), which generates adhesive forces via a vortex inside of the robot. This vortex is created by a high-speed rotor and allows adhesion without additional sealing elements as needed by the cup-based systems. So far only a few systems have been developed using this principle as e.g. the Clarifying Climber Robot by Clarifying Technologies, alicia vtx by [Bonaccorso2009] or the entertainment robot ParaSwift [Geissmann2011], which are all wheel-driven and shown in figure 2.23. A basic platform called vmrp (vram Mobile Robot Platform) can be bought for about 13 000 USD15. A similar setup for locomotion has been chosen by [Journee2011] for a robot using the venturi effect to generate the needed negative pressure (figure 2.23d).

![Figure 2.23: Climbing robots using the vortex mechanism (a–c) or the venturi principle (d) for adhesion in combination with wheels or tracks.](image)

Passive suction cups are suitable only on very flat surfaces like glass, but can also be combined with different types of locomotion like the tracked vehicle by [Yoshida2010] or the dexter robot by [Brockmann2004], which is equipped with two articulated feet.

This principle has the big disadvantage that it is not robust against disturbances like dust. Furthermore, that the robot must stay in motion, since the cups slightly lose negative pressure because of small leakages. Therefore, the most common approach is the usage of active suction chambers in combination with an electrical vacuum generator for evacuating the adhesion cups. In general, these generators either produce a large through flow volume or a high negative pressure – depending on the robot’s setup. Known wheeled systems are e.g. WallWalker [Miyake2007], BIT Climber [Li2010], ALICIA3 [Longo2006], LARVA [Song2008] or the robots by [Qian2006] and [Yan1999] (see figure 2.24).

Examples for robots using sliding frame mechanism are the different Sky Cleaner systems [Zhang2006a, Zhang2009], PlanarWalker [Chen2003] or the machine by [Gradetsky2012] (figure 2.25). Legged systems have been developed by [Luk2005, Luk2006, Luk2006a] in terms of the different ROBUG and NERO robots equipped with active suction cups at their feet as well as the systems by [Tummala2002, Hong2009, Liu2011]. [Kim2010] introduced a tracked robot consisting of several arm-like components. Additionally, there exist systems like SkyBoy [Wang2010a], which are moved by a belt or tether supported [Qian2006a] for cleaning glass facades. A less common locomotion type has been used by [Hayakawa2009] whose robot performs a kind of snail-motion (figure 2.25l). The main problems related to negative pressure adhesion are the leak tightness of the suction cups especially on rough surfaces and a high power consumption if active vacuum generators are used. Here also legged systems have the advantage that they can test the leak-tightness of each foothold and to test other positions if the adhesion is not ensured. On the other hand these systems are relatively slow and Therefore, not applicable for common tasks, which require a sufficient fast navigation speed.
Another adhesion principle is mechanical adhesion based on claws (or spines) or via a gripping or clamping mechanism, as depicted in figure 2.26. Climbing robots equipped with (micro-)spines like rise [Spenko2008], SpinyBot II [Kim2005], or clibo [Sintov2011] are commonly legged systems having the spines at their feet. These robots often have a simple mechanical structure and a low weight, which enables them to climb even on concrete walls. Unfortunately they are not suitable to carry a high payload of several kilogram. The robot LEMUR IIb is also equipped with legs, but it uses existing protrusions for climbing [Bretl2004]. Another principle of locomotion has been introduced in terms of the robot ROCs [Provancher2010] sticking to walls coated by a carpet or other net structures. This robot moves up the wall by swinging a mass from left to right shifting the two claws upwards alternately. In contrast to that, there exists a couple of robots like CliBot [Schober2010] or ROMA [Abderrahim1999], which use arms with clamping or gripping mechanisms to stick to the surface. Of course, there must exist a corresponding structure with suitable points of adhesion. ROMA e.g. has been designed to climb on
a beam-based structure with its two grippers. But, there also exist tracked or wheeled systems like MovGrip [Chung2011] pressing their motion units to a pole or a wall by enclosing it. For climbing on concrete walls some systems are applicable (e.g. RISE robot) whereas others are specialized to certain structures. Especially the clamping and grasping mechanisms are very safe with respect to robot adhesion, since they can be realized in a way that the grip closes in case of a blackout. In contrast to that, robots using spikes strongly depend on the surface structure, which makes a drop-off possible.

![Figure 2.26: Climbing robots using claws and spikes (a–e) or grippers (f,g) and clamping mechanisms (h).](https://example.com/robot-climbing)

A growing discipline in the research field of climbing robots is the **electroadhesion**. Here electroadhesive pads comprising conductive electrodes and insulation substrate are used to generate electrostatic or Van der Wals forces between the surface and the robot. So far there only exist a few of robots using this active adhesion mechanism like the tracked vehicles by [Prahlad2008] and [Cui2012], as shown in figure 2.27. Another kind of electrostatic force is the passiv gecko principle based on microscopic hairy structures, which adhere to the surface on an atomic level. Here tracked vehicles [Greuter2006, Menon2004] can be found as well as legged robots by [Unver2006], Waalbot by [Aksak2008] and StickyBot by [Kim2007].

Beside the presented principles there exist a large variety of additional adhesion approaches such as **sticky tapes** [Daltorio2005, Seo2011] or **thermal glue** [Rochat2011]. In general most of these systems are specialized to certain environmental setups and tasks or have just been developed as a proof of concept. With regard to the application of inspections of concrete buildings the gecko principle might not be the optimal solution due to a low payload. The same counts for the systems using sticky tapes. In contrast to

---

17http://unews.utah.edu/news_releases/robot-climbs-walls/
that, active electroadhesion seems to be a very promising approach. Unfortunately this technology is not really completely understood and still at a first stage of development. Nevertheless, it is possible that it will outperform the other approaches within the next ten years.

Figure 2.27: Electroadhesive climbing robots with active adhesion (a,b), the gecko principle using microscopic fibrillae (c–g), and robots using less common techniques like sticky tapes (h,i) or thermal glue (j).

The introduced systems and principles are without any claim of completeness. But, this section should give a short summary on existing approaches and on their manifoldness to get an idea of their fields of application and of current research aspects. More information on the presented adhesion techniques, locomotion types, and robotic systems can be found in several further publications [Chu2010, Longo2008, Silva2008].

2.3.2 Comparison and Discussion

A main requirement for all these systems as well as for any other robot is safety, which is generally related to a drop-off of the system in the range of climbing robots. This especially addresses malfunctions of the adhesion system and external disturbances e.g. during navigation because of changes in the surface structure (assuming a more or less static world in the case of climbing robots). Of course, there is always a tradeoff between applicability and safety, since – in general – safety measures limit the capabilities of a system or the field of application\(^{19}\).

Very safe systems in the field of climbing robots – able to navigate on concrete walls – are those, which can test their adhesion at certain ground positions before they continue their navigation. An examples for this is the fourlegged robot Robug II (figure 2.25e) by Bing Luk et al. from the City University of Hong Kong \[Luk2005\]. Robug II can be equipped with vacuum grippers at the feet and with adhesion units at the underside of its body. By this mechanism the system is able to move like a sliding frame robot and either adhere the main body and move all four legs in parallel or vice versa. This allows a faster movement than moving each leg separately. For adhesion the suction chambers reach a pressure of about 80% of the ambient air, which corresponds to an absolute pressure of 80 000 Pa. Compared to CROMSCI this system is more flexible and is able to perform transitions from ground to wall and to overcome larger steps. Nevertheless, it is not clear how fast the system is and if the needed support units are operational in the field, since the legs are moved via pneumatic actuators. Again, all legged systems using active negative pressure adhesion like Robug II or the machine by \[Kim2010\] can test adhesion points and are not exposed to unforeseen disturbances. Only a spalling of the surface might become problematic as in the case of all other climbing robots.

A similar approach compared to CROMSCI has been developed by Wang Yan et al. from the Harbin Institute of Technology, as shown in figure 2.24h. This robot uses an omnidirectional drive consisting of four wheels reaching a speed of up to 8 m/min (0.133 m/s) \[Yan1999\]. With a weight of 30 kg it nearly plays in the same league as CROMSCI although its dimension of 0.29 m × 0.26 m × 0.23 m is smaller. Even the maximal payload is stated to be about 10 kg. Unfortunately, there exist no further publications on this interesting system and it disappeared from the scientific scene. Instead of that the authors focussed on window cleaning robots, as depicted in figure 2.24g. The main difference to CROMSCI is that it has only one single vacuum chamber, which allows no force balancing and seems to be a single point of failure in case of any unpredicted disturbance of the adhesion system. This aspect might be the biggest drawback of this approach.

The same counts for the much lighter climbing robot of Toru Hayakawa et al. from the Chuo University in Japan depicted in figure 2.25l. Their system also consists of a single adhesion chamber evacuated by fans \[Hayakawa2009\]. Although the system has similar dimensions (0.29 m × 0.315 m × 0.15 m) compared to the previous robot its weight is only 2 kg. The sealing is realized via a brush and it uses a wave propagation unit to perform snail-like locomotion. Depending on the surface structure the system is able to reach velocities of 0.84 m/min (coarse tiles), 1.38 m/min (smooth surface) up

\(^{19}\) A common example is a safe computer, which should not be infected by malware or a virus. In this case one has to cut the internet connection and remove all ports and devices for removable media, which makes the system safe, but not very handy.
to 2.52 m/min on grooved tiles in vertical direction. Experiments have shown that this system is able to climb even on walls with a lot of gaps. It is assumed that the low weight and the – compared to that – large suction area lead to this performance, since the needed negative pressure for adhesion is low. Nevertheless, there are no redundant chambers as in the case of the presented legged systems or CROMSCI. There is also nothing said about the navigation capabilities and the possible payload, which both are important for task fulfillment.

It should be obvious that climbling robots using negative pressure adhesion are suitable for the inspection of large concrete structures, but still not safe enough. Other adhesion types – e.g. spikes and sticky tapes – are not capable to carry the needed payload or still in exploration phase like the principle of electroadhesion. The main challenge is to find a good compromise between operability and operation safety. Here it has to be considered that safety is very important but limits the operability of the system. The next chapter will introduce common approaches to identify single points of failure or safety critical elements. Additionally, causes for a robot drop-off in the case of wheel-driven climbing robots using negative pressure adhesion will be discussed.
3. Hazards, Risks and Safety Aspects

The assessment of risks and considerations concerning safety have been studied since the very beginning of modern machines. Nevertheless, the definitions of risk and safety are strongly connected to the point of view. In this chapter aspects concerning safety will be presented and in which way safety and hazards can be identified, analyzed and handled. At first, the kind of safety will be described, which is taken into regard here. Therefore, this chapter starts with some definitions concerning safety and related terms in section 3.1. Also the distinction to other definitions will be given here. After that, important aspects concerning safety guidelines follow in section 3.2. This includes international standards, which are applied in industry, up to considerations concerning the so-called self-safety of mobile robots. Section 3.3 presents approaches for safety analysis and for risk assessment. The final section 3.4 contains descriptions and a classification of hazards for climbing robots like CROMSCI.

3.1 Definitions

Before introducing methods of risk detection and safety analysis, different definitions have to be given concerning the desired context within the scope of this thesis. This is important because of the broad range of application areas and definitions of these terms.

Hazard

As given in definition 1, a hazard is a situation with a potential to undesired effects like damages or injuries. If a hazard occurs it is called incident. The iec 61508-1 defines a hazard as a “potential source of harm” [IEC61508].

Definition 1: Hazard

A hazard is a situation, which could lead to damages (with a certain probability and severity). In general, it is armed, if the system operates and it is dormant if the system is not in use (if the situation cannot occur).
In the case of climbing robots, a hazard might be a collision between the robot and an obstacle or the loss of adhesion, which could result in a total loss as given in definition 2.

**Definition 2: Hazard for climbing robots**

A hazard for climbing robots is a situation, which could lead – with a certain probability – to a drop-off of the system or to serious damages. This includes also incidents, which become dangerous only in combination with other events.

Unfortunately, there exists an unlimited number of failures, aspects, and influences leading to damages. Therefore, the identification of external impacts and system subcomponents posing a hazard is very important. Beside the external influences and situations the developer has to respect the complete mechanical structure of the system, data busses, interfaces, power supply, electronics, sensors and motors, controllers, software components and much more. The hazards, which might occur, can be classified related to the severity of the threat and the effect on the safety requirements. Table 3.1 gives an overview on five severity classes used as hazard classification scheme as proposed by the European Organisation For The Safety Of Air Navigation, which is suitable for most technical systems [Eurocontrol2000]. This scheme does not include any probabilities for the occurrence of the individual hazards.

<table>
<thead>
<tr>
<th>severity class</th>
<th>effect on operations</th>
<th>results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (most severe)</td>
<td>complete loss of safety margins</td>
<td>catastrophic accidents, no source of recovery</td>
</tr>
<tr>
<td>2</td>
<td>large reductions in safety margins</td>
<td>serious incidents, avoidance actions to avoid catastrophe</td>
</tr>
<tr>
<td>3</td>
<td>major reductions in safety margins</td>
<td>major incidents</td>
</tr>
<tr>
<td>4</td>
<td>slight reduction in safety margins</td>
<td>significant incidents, only indirect impact on safety</td>
</tr>
<tr>
<td>5 (least severe)</td>
<td>no effect on safety</td>
<td>no hazardous condition</td>
</tr>
</tbody>
</table>

In the scope of this thesis, a catastrophic accident with severity class 1 is related to a drop-off of the robot, which would destroy the system and maybe injure a person standing beneath. The avoidance of situations leading to this highest severity is the main requirement and challenge for climbing robots.

**Risk**

The term *risk* is hard to conceptualize because various definitions are given for different applications. Holton tried to give a very general definition: “*Risk (...) is exposure to a proposition of which one is uncertain*” [Holton2004, p.22]. According to his explication
risk requires an amount of uncertainty and an exposure. He gives the example that a person trying to attempt suicide by jumping off from a high place faces no risk of dying because he is certain about the consequences. Actually it can be said that this person is confronted to the risk of surviving this attempt strongly injured – thus the general definition of risk is not necessarily linked to hazards or damages.

Related to technical systems, the standard IEC 61508-1 defines risk as follows: Risk is “a combination of the probability of occurrence of harm and the severity of that harm.” [Ladkin2008, p.3]. The type of this combination between harm and probability differs. A common mathematical declaration is that risk is the probability of an accident multiplied by the estimated amount of damage [Daneshkhah2004, Ertle2010]. Kaplan et al. state that risk is probability and consequence [Kaplan1981]. Otherwise a scenario with low probability and high damage would be equal to a scenario with high probability and low damage – which is not the same at all. Definition 3 gives a general description of a risk:

**Definition 3: Risk**

A risk is the probability of a hazard (with a certain severity) to occur.

In the present case, the most endangered object is the climbing robot itself: A loss of adhesion would lead to a total loss of the system. Therefore, the following specialized definition 4 is used within this context.

**Definition 4: Risk for climbing robots**

A risk for climbing robots is the probability of a hazard to occur, which would result in severe damages or a total loss (drop-off) of the system. It is assumed that no internal effects can lead solely to such a black out. Only external effects (in combination with the internal system state) have an influence on risk for climbing robots.

To distinguish between the terms hazard and risk for climbing robots it can be stated, that a hazard exists as a source of danger with a certain accident severity whereas a risk is the probability of a hazard to occur.

**Safety**

In literature, several descriptions and definitions of the term safety can be found. A very common definition is: “Safety is the condition of being protected against failure, breakage, error, accidents or harm. Protection involves here both causing and exposure” [Kojima2005, p.5]. In contrast to this very strict definition, safety can also be defined as “freedom from unacceptable risk” [Eurocontrol2009, p.65]. Here, safety is seen as the control of hazards, to reduce the risk to an acceptable level (see definition 5). Other definitions implicate that there exists no total safety and that the safety itself is relative to the possible hazards and the application.

1http://www.businessdictionary.com/definition/safety.html
Definition 5: Safety
Safety is the reduction of risks to an acceptable or tolerable level to achieve a relatively good protection against hazards.

Depending on the application of the technical system the imperiling influence from external sources might increase. As it will be shown in section 3.2.1, the well-known and well-defined environment of an industrial robot decreases the complexity of safety analysis and allows the usage of several existing techniques to analyze hazards and ensure (human) safety. In mobile robotics and especially in the field of climbing robots, the robot components are assumed to provide a relative safety compared to the unknown external disturbances as given in definition 6. Here hardly predictable situations and influences exist because of missing world knowledge, a lack of exact physical models, or missing methods to detect them.

Definition 6: Safety of climbing robots
Safety for climbing robots is the reduction of risks to an acceptable or tolerable level to achieve a relative protection against hazards caused by external influences. The system components themselves are assumed to be relative safe compared to the possible effects from outside.

Based on the given definitions, the next section will present necessary aspects which need to be taken into account for safety requirements and show, how the safety of robots and other safety critical systems can be guaranteed or at least analyzed and enhanced.

3.2 Safety Guidelines
“Safety critical systems are those systems that can potential lead to loss of life, injury and environmental damage” [Wilson1995, p.765]. Since robots interact with their surroundings in terms of human operators or work pieces, they can be added to the collective term of safety critical systems. These systems should not only provide the desired functionalities, but also meet certain safety requirements to minimize the threats caused by them. For classification and analysis of desired safety aspects for robots, Ward and Went characterize three different safety levels in a much cited article, as illustrated in figure 3.1 [Ward1995]. These levels point out, how large a risk for a human operator or the robot’s environment will be. The authors use this scheme to describe, how the development process of a robot system should be performed to fulfill the necessary safety requirements. They point out that it is important to take the whole life cycle of a robot into account – including design, installation, use and maintenance – to identify hazards.

Despite all modelling, analyzing and improving it has to be kept in mind that no robot will be totally safe and a remaining risk will always exist. In general, the threat can only be
3.2. Safety Guidelines

Figure 3.1: Three levels of risks for safeguarding, as described by [Ward1995].

redced, but not eliminated. Thus, the principle of as low as reasonably possible (ALARP) is applied in many cases\(^2\) while it is also requested by law for industrial machines, e.g. in the Health and Safety at Work etc. Act 1974 [HMSO1974]. Beside the technical component also human faults have to be considered if both, human and robot, act collaboratively, as Hirochika Inoue from the University of Tokyo points out: “No matter how sophisticated your robot is at avoiding people, people might not always manage to avoid it, and could end up tripping over it and falling down the stairs”[\(^3\)].

In the following sections considerations, rules, and requirements for safety of robotic systems will be presented, starting with well-known industrial robots. This is important to get an idea of the complexity of the safety requirements and the challenge of their realization in the field of climbing robots.

3.2.1 Safety Requirements in Industrial Environments

Industrial robots were first established in 1960s in automobile industry and are used till now for different automation tasks [Ichbiah2005]. Of course, safety is a necessary requirement for the one million industrial robots worldwide [UNECE2002]. But, if there is talk of the safety of an industrial robot, in nearly all cases the protection against damages of the robot environment or against injuries of persons is meant. The reason for this is the well-known mechanical structure and functionality of the robotic manipulator and its fix position: In general, industrial robots will cause damages only to the environment and not vice-versa.

Because of the threat caused by industrial robots the International Organization for Standardization (ISO) formulated the international standard ISO 10218-1 in 2006 dealing with

---

\(^2\) In literature one can find further methods of risk accepting like globalement au moins aussi bon (GAMAB) or minimal endogenous mortality (MEM), which e.g. take existing systems or the overall mortality rate into account.

3. Hazards, Risks and Safety Aspects

safety demands for these systems [ISO10218]. In this document different requirements
and hazards are formulated, which describe what kind of risk assessment shall be carried
out and in which way the robot should be designed regarding relevant hazards. Most
requirements and protective measures deal with situations in which the robot does not
cooperate with humans. This includes e. g. a loss of power, malfunctions of single compo-
nents, protection from unintended operation, or the indication of the current system state.
For the collaborative mode – in which a technician acts within the same workspace as the
robot – the international organization formulates more strict requirements to improve
the safety of the robotic system. Some of these requirements taken from [ISO10218, sec.
5.10.3 - 5.10.6] are:

- Power and force must be limited to a maximum dynamic power of 80W or a maxi-
mum static force of 150N at the tool center point.
- Limitation of velocity to maximum 0.25m/s if a human is inside of the range of the
robot.
- Reduced speed if the robot is hand-guided. Otherwise the position of the human
must be monitored to initiate an emergency stop if he gets too close.
- Protection stop if any maximum value is exceeded.

Of course, these systems are not that safe in a way that they could not harm a per-
son. Especially collaborative operations include collisions between the working partners.
Therefore, the effects of these in some cases tolerable, but unwanted collisions must be
reduced to an acceptable injury severity. In general, a standardized safeguarding process
is applied according to the international standard IEC 61508-1 consisting of three steps:
The risk assessment contains methods of risk analysis and rates, whether a tolerable risk
level has been reached or not. While the risk is not tolerable the engineer has to apply
safety measures to lower the risks. Finally, safety measures by the user need to be realized
in terms of safety instructions, regular inspections or employee training. A more detailed
description of requirements for collaborative working as well as information about the
standardized safeguarding process can be found in appendix section A.5.

3.2.2 Safety Guidelines for Mobile Robots

Compared to mobile robots the job of analysis and implementation of safety aspects is
much easier in the case of industrial robots. Their tasks and behaviors, but also their
working environments, are well-defined, which makes it simpler for robotic developers
to implement the desired functionalities and to make the system (more or less) safe. In
mobile robotics the world is a priori unknown. For a robot developer, it is impossible
to consider all events sufficiently which could occur in the dynamic environment the
robot is located in – particularly with regard to the methods for safety analysis relying
on these considerations. Or as Robinson points out: “While this adaptability makes a
programmable robotic system desirable, it also makes the safety analyst’s job that much
tougher” [Robinson1996, p. 5].

Since the beginning of mobile robots the developers considered safety aspects. At first,
they used simple electronics like switches to cause an emergency stop. First research
aspects concerning mobile robot safety have been path optimization and evasive actions. The challenge was—and still is—to identify a hazard and to find an appropriate manner to react on it. Over the time, additional safety measures in hard- and software have been developed using a wide range of techniques (e.g. genetic algorithms [Chen1995], fuzzy logic [Zurada2001] or potential fields [Shimoda2005]). The final goal is to ensure safety while the robot—e.g. a humanoid like Honda’s ASIMO [Sakagami2002]—interacts with a person.

Unfortunately, there exists an unlimited amount of components, technologies, possible failures and hazardous situations, which have to be taken into account. Currently a consortium of different robotic scientists is working on the new international standard ISO 13482, which deals with safety aspects for non-medical personal care robots like mobile servants, person carriers and physical assistant robots [Virk2010]. The goal of the consortium is to formalize analysis methods and development steps to restrain the problem of increasing safety requirements and system complexity. As other standards before, ISO 13482 will be limited to human health-related hazards and will regard neither environmental nor robotic damages. Nevertheless, an appropriate risk analysis is needed again although the standard will not be able to provide detailed metrics because of the complexity of the systems and their non-standard working environments. Therefore, ISO 13482 will only contain detailed development guidelines to be followed. The robot manufacturer has to perform risk analysis for the specific robot and to demonstrate the necessary hazard identification measures and protection methods. For the evaluation of risks the existing machine standards ISO 12100 and ISO 14121 provide details for determination of robot limits, hazard identification, risk estimation, hazard elimination and risk reduction [ISO12100, ISO14121].

### 3.2.3 Self-Safety for Autonomous Robots

All the international standards and guidelines considered hazards for the environment caused by the robot. But, of course developers also have to think about the self-safety of these systems. Especially in the case of climbing machines the robot itself is the most endangered object and the environment can be neglected. In the range of this topic, Kelly and Stentz from Carnegie Mellon University defined some criteria, which are important for self-safety of autonomous mobile robots [Kelly1998]. They describe four aspects for a policy of guaranteed safety for autonomous systems, which act in an a priori unknown world:

- **Detection** The robotic system must be equipped with sensors and use methods to detect and analyze every environmental feature, which could harm the mechanical structure of the robot.

- **Localization** It has to use methods to locate the detected environmental features accurately with respect to itself, otherwise the feature information are useless.

- **Response** The robotic vehicle must react fast enough and in the correct manner to the detected and localized features to avoid the detected hazard.

- **Throughput** The sensor update, data extraction, and the update of the internal world model must correspond to the velocity of the robotic vehicle.
All these aspects meet questions regarding the performance of the overall system, which are also related to individual components. In contrast to the previous considerations these aspects address only functional requirements. If a component operates as desired it is not assumed to fail at any time (in contrast to the approach of [Guo2010]). Kelly and Stentz declare their robot safe if all four aspects are considered. Unfortunately, almost no robot fulfills all aspects in general. But, for specific tasks and describable environments a guarantee for safety can be given. Therefore, the world with all objects and physical processes, especially the robotic system itself and its physical reactions with the environment have to be known sufficiently. This does not include an existing world model, but the knowledge about environmental features the robot might be confronted with. As presented in [Kelly1998], there are a lot of elements, which have to be taken into account for autonomous vehicles: The awaited positive and negative obstacles, the vehicle speed and its general setup, but also sensor arrangement, sensor resolution and update rate. A small-sized robot for example might be endangered by tiny obstacles, which are harmless for larger vehicles. Here the sensor resolution must be higher to detect these small features, the update rate might be smaller because of a lower robot velocity.

Figure 3.2: Examples describing the relationship between harmful environmental features and vehicle parameters [Kelly1998].

Examining autonomous ground vehicles, there are different environmental features which either have to be taken into account because of robot geometry or which can be neglected. All overhanging obstacles located above the robot cannot harm it, obstacles on the ground have to exceed a certain size, which is e.g. depending on the wheel size, wheel distance, and ground clearance to become dangerous for the vehicle (figure 3.2). In general, the potential risk caused by specific features can be described if it is known, how these features interact physically with the robotic system. It is obvious, that – given the certain vehicle parameters – there exist obstacles and holes, which cannot be passed by the robot. These elements have to be avoided because the effect on the robotic system is clear: Either its structure will be damaged or it becomes stuck in an unsuitable location. Similar considerations about the influence of different terrain aspects and their impact on a mobile robot have also been presented by Henriksen and Krotkov one year earlier [Henriksen1997]. Of course, these approaches only consider hazards which are expected and which can be described sufficiently. Nevertheless, the description and handling of obvious hazards is not sufficient for safe navigation of a climbing robot. In contrast to common ground-based vehicles climbing robots have to deal with some hard to predictable situations and events,
which could lead to a total loss of the system. This addresses mainly the detection of those surface features, which may have an influence on the robot’s adhesion, but also the correct response of the system. Demands to achieve a tolerable safety level for climbing robots to perform the desired navigation tasks will be introduced in chapter 4.

3.3 Safety Analysis and Risk Assessment

The safety guidelines from previous section 3.2 give an idea, in which way the safeguarding process during development phase and runtime can be handled. Nevertheless, this scheme does not provide any details or instructions how hazards can be identified or what kind of system adaptions need to be applied to reduce the severity or the probability of a certain hazard. In fact, these two aspects are a main challenge in the range of mobile robotics and especially in the case of climbing robots.

This section will give attention to the safety analysis and presents some methods, which are commonly used in the range of technical systems to detect potential hazardous situations or error states and their causes. The proper handling of the hazard strongly depends on the system and the environment in which it is operating and will be handled in terms of some examples from literature. In the field of mobile robotics, it has to be considered that safety analysis includes offline analysis during robot development and design as well as a monitoring during runtime. In literature a lot of different safety analysis methods for robotic systems can be found, which go back to the very beginning of robot development [Dhillon2003].

3.3.1 Standard Analysis Methods

In general, three phases of safety analysis can distinguished as proposed by [Wilson1997]. At first, hazard identification techniques like hazard and operability studies (HAZOPS) [Sahar2010] or functional failure analysis (FFA) are used to identify the potential hazard for a system or a subcomponent. After that, the causal analysis determines potential causes and an estimated likelihood of each hazard, e.g. via fault tree analysis (FTA). In the last step – the consequence analysis – the potential accidents, which could arise from the determined hazards, and their likelihoods are identified and rated using techniques like failure mode and effect analysis (FMEA). The results of these analysis methods are then used to prevent the occurrence of incidents or to reduce the risks as low as reasonably practical. To get best results usually more than one safety analysis technique is used. A short summary of useful analytic methods is given in appendix section A.5.2.

General analysis methods have in common, that they either use colloquial speech to describe hazards and their causes (qualitative description) or need statistical data about failure rates or the likelihood of actions (quantitative description). In general, the cause and effect of an industrial robot action can be described well like the consequence of a collision between robotic manipulator and a human [BGIA2009]. The main problem is to

4Beside the technical system itself some analysis methods concerning industrial robot and machine safety additional take the human operator and his actions into account. In general, this has to be considered because of potential mistakes made by a human operator. An even bigger problem arises from reduced comfort or time pressure: In 37% of the industrial companies easy accessible protection devices like safety or service switches are unused or bypassed because of these reasons [Umbreit2010].
create a model of the overall system behavior, which satisfies a certain precision to get meaningful analysis results. Human behavior e. g. cannot be modelled explicitly by most techniques. Other methods like the markov analysis need an explicit identification of the system states and all possible transitions or, in the case of dynamic event tree analysis method (DETAM), a large data collection as a basis for the analysis.

Table 3.2 sums up basic analysis techniques (described in section A.5.2) and states, which methods are generally applicable for this special purpose. Often a combination of different methods is used to get meaningful results. By default, the specialized methods concerning the software system, human actions, or the interfaces cannot be used here. Also the markov analysis is not applicable because it needs reliable occurrence rates of failures, which cannot be collected. The TOR method (technic of operations review) does not satisfy the requirement because it allows an analysis only after the failure has occurred and not in the design phase or online. But, some of these techniques like risk graph method, FMEA, or fault tree analysis have been used in the range of mobile robotics, as shown in the next section. In the present case, those techniques are useful, which deliver information about the causes and underlying effects like CAED (cause and effect diagram) or fault trees.

3.3.2 Related Work in Risk Assessment and Handling

The introduced analysis techniques are not only of theoretical importance, but also applied in the range of mobile robotics. This includes both offline analysis as well as online (or dynamic) analysis methods for risk assessment. Offline methods – as they have been introduced before – are mostly used in the design phase or after an incident has happened. The second category contains methods to determine online risk values for mobile robots, which interact with a dynamic environment. But, beside the analysis approaches also the corresponding measures for risk handling differ. While in design phase the developer has the option to use hardware measures as well as software elements to increase the safety of the system. During operation there is no way to add components. This is only possible offline, e. g. after an accident has occured. Then analysis methods like CAED or TOR are used to identify the cause of the error and additional safety measures are applied. Online measures have to be implemented offline, but might be adapted by the system online during runtime. In general, safety-related control or protection devices use so-called electrical / electronic / programmable electronic (E/E/PE) technology like safety chains, relais or digital signal processors (DSP). The whole challenge in this area can be summed up to “find a risk and handle it”.

Researchers from the University of Kaiserslautern presented a scheme to identify requirements concerning safety and security for the outdoor robot RAVON [Guo2010]. The authors Guo, Zeckzer, Liggesmeyer and Mäckel use analysis techniques like FMEA and FTA to detect internal malfunctions of individual components, which could lead to a black-out of the system so that people may get hurt. Additionally, they combine branches of a security attack tree into the safety fault tree to get a complete description of each component regarding safety and security requirements. Using this approach the authors are able to identify these requirements as it would be helpful also in the scope of this thesis. Nevertheless, this is only one part of the whole safeguarding process and one needs to know more about the error sources e. g. in terms of MTTF or other malfunction probabilities to get a more detailed statement in the present case.
### 3.3. Safety Analysis and Risk Assessment

Table 3.2: Overview on the presented analysis methods and some characteristics which are either satisfied (●) or not (○).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>General Attributes</td>
<td>---------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Qualitative</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Quantitative</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>•</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Causes</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Universal</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Probability Data</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>System Knowledge</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Special Data</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Application</td>
<td>---------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Risk Assessment</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Fault Analysis</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Design Phase</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Applicable</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
</tbody>
</table>

*a* The *general* attributes show, if the methods use qualitative (e.g. 'severe damage') and/or quantitative data (e.g. '1% chance of failure'), if they provide information about the causal sources of failures, and whether the method is universal for different kind of systems or not.

*b* This section lists, what kind of knowledge is used or can be processed by each method. In the last case (special data) software components are examined.

*c* The *application* part shows, for what kind of analysis the techniques can be used. A couple of them provides risk values or classification, others enable an analysis of failure sources and underlying events. About half of them can be applied in the design phase of a system.

*d* This row shows if a method is applicable generally for this purpose or not. This does not mean that it provides useful information for risk assessment and handling of a climbing robot.
Fault trees are also used by Robinson and Atcitty from the Sandia National Laboratories. In a first step, the authors use a variation of FMEA to estimate the potential risk of a so-called high consequence robotics system working in nuclear plants or other safety-critical environments [Robinson1996]. Beside the traditional issues like personnel safety also other factors like the sensitive handling of radioactive payload have to be considered here. The idea of Robinson and Atcitty is to break down the entire robot process into smaller steps, which can be handled and described easier. Each of these steps is analyzed independently using FMEA and the results are combined via a fault tree. Based on this, it is possible to set up a formal description using boolean algebra to calculate a rating value of the risk. This analysis is done for each process step to get an evidence concerning the corresponding hazard level. In an example the authors describe the simplicity of the final calculation. They use three events seizure of motor or reducer \((F)\), overcurrent noted by control software \((D)\) and compensation of detected overcurrent \((C)\), which have an influence on safety. A fault tree links these events – each of them equipped with a certain probability \(P()\). Finally, a formula as given in equation (3.1) can be set up. In colloquial speech an accident occurs in case of a seizure of a motor or reducer and if either the overcurrent is not noted by the control software or it is noted, but not compensated. By defining probability ranges for each event the upper and lower bounds on the likelihood of an unhandled failure can be derived.

\[
P(\text{accident}) = P(F) \cdot ((1 - P(D)) + P(D) \cdot (1 - P(C)))
\] (3.1)

It is obvious, that this analysis strongly depends on reliable failure data. As the authors point out, they use failure rates from the manufacturer, which are based on service calls concerning similar operating systems and thus increase the uncertainty of the results. The transfer of this approach to the climbing robot cromsci fails because of two aspects: There neither exists reliable failure data nor is it possible to define a manageable amount of states. As pointed out before, leak-tightness, wheel abrasion, and other factors strongly depend on the characteristics of the surface. This interaction between the robotic system and the environment cannot be described and modelled sufficiently. It is also not possible to record enough reliable data to get even a rough description of failure rates.

Another way of risk assessment has been discussed by Tiruchinapalli from the University of Applied Sciences Bonn-Rhein-Sieg. He presents safety concepts for mobile robots by taking hazard examination, risk analysis, and safety functions including their effectiveness and correctness into account [Tiruchinapalli2005]. According to the standard IEC 61508, he illustrates different measures and decisions in hard- and software design to ensure an acceptable level of risk. As a demonstrator, he uses the small ground-based vehicle KURT3D applicable for rescue scenarios. At first, he describes hazards like crushing, friction, slipping or failure of the energy supply. Consequently, a risk value for each hazard is estimated depending on severity \((C)\), frequency \((F)\), probability \((W)\) and avoidability \((P)\). He uses the risk graph method (see section A.5.2) to perform a qualitative analysis of the risks. Figure 3.3 illustrates the results of the analysis of the crushing hazard. According to IEC 61508 the following classification of the risk parameters has been determined: The consequence could be serious and permanent injury to people \((C_2)\), the robot is permanently exposed to the risk of a crash \((F_2)\), in general it is possible to avoid a crash \((P_1)\)
and the frequency of unwanted occurrences would be slight ($W_2$). In this example the result is a SIL of 1, which makes some special requirements necessary. In this context he points out, that SIL is a statistical degree for the integrity of a safety system and that “a chain is only as strong as its weakest link” [Tiruchinapalli2005, p.43]. There is no reason for enhancing safe elements while others are ignored. To ensure the operability of the safety measures Tiruchinapalli examines the European norm EN 954-1 [EN954-1], which requires self checks of the safety device. KURT3D e.g. surveys the motor controller via a watchdog unit and executes an emergency switch off if the controller fails. Additionally, sensor values and the watchdog itself are tested by the microcontroller for invalid states by analyzing their output values. If the microcontroller is operating normally, but the watchdog has a malfunction, this is noticed by the controller. The desired behavior is that the robot enters a safe state – in this case a switch off of the drive units – if any failure is detected successfully. In the final discussion he pointed out, that the early detection of faults is not sufficient solely. Also the reactions of the safety devices have to be correct in a way that they lead to a safe state. The problem regarding a climbing robot like CROMSCI is based on this definition of a “safe state” and – again – on the system knowledge. A safe state is reached by a mobile robot in general if it stands on flat ground without moving. Of course, this will not happen for a climbing robot during normal operation. Therefore, there exists no safe state in a strict sense since it is not said that the robot will adhere safely at its current position. Beyond that it is not easy to define aspects of a safe state if the interaction of robot and environment cannot be modelled. Here some system adaptions by fuzzy or learning methods are required to identify risky and more safer states.

In the range of mobile robotics, not only the internal machine state and components must be surveyed and examined. Especially the safety analysis concerning the interaction between environment and robot is of growing importance as it is considered by Wardzinski. The researcher from the Gdansk University of Technology describes a method for risk estimation of current and foreseen situations of autonomous vehicles [Wardzinski2008]. He points out, that – given different risk levels – the main objective is to prevent transitions from one state to a more unsafe state. The basis for the estimation of risks and corresponding reduction possibilities is situation awareness. In his work he uses three levels of human situation awareness: Perception of the environment, reasoning to combine in-

![Figure 3.3: Risk graph method](image-url)
formation and to decide if they are useful for reaching the goal, and *prediction* of future events and their influences on the system. The idea is to check different options of reaction if a risky situation is predicted and use the safest one. As an example, he uses two robots with specific trajectories, which would meet in an intersection point as shown in figure 3.4. The risk factor is the predicted minimal distance with the assumption, that the second robot does not change its trajectory. For risk classification, he defines four risk levels reaching from A (no risk), B (acceptable safe) up to C (hazardous) and D (accident). By a specification of qualitative values for each level the control system is able to estimate risks for a set of trajectory adaptions. Here certain distance values are used as measuring unit. Adaptions could be e.g. turning or a change in velocity up to a full stop. Depending on the balance of goal-heading and risk-safety it is possible to get a more aggressive or a defensive vehicle, which either tries to reach the target as fast as possible or minimizes the risks of its further motion. Wardzinski also tries to include uncertainty values concerning the behavior of the second vehicle by adding it to the situation awareness model. Unfortunately, the results are not satisfying without a mutual agreement of the robots about their routes, but they show the capability of this approach. In his summary the author points out that the main problem in the usage of a situation awareness model is the “*completeness and accuracy of the situation model to adequately represent possible situations*” [Wardzinski2008, p.130]. In other words: An exact vehicle model is needed as well as an accident model representing the mechanisms of occurrence and prevention of accidents. In the context of mobile robotics, the accident model needs to contain the description of the interaction between robot and its (dynamic) environment. This knowledge can be achieved more or less easily for standard ground vehicles, but it is – again – almost impossible for dynamic climbing robots using negative pressure adhesion.

![Figure 3.4: Example of a trajectory crossing of two robots as given in [Wardzinski2008]. Optional trajectories are tested against an estimated risk value for that route.](image)

A more comprehensive view on safety of robots is given by **Ertle, Voos and Söffker**. Beside the standard and mostly passive safety measures, which are applied in the development phase of the robot, some additional active measures are needed to ensure safety while it is operating [Ertle2010]. The authors mention three main elements to avoid failures: *Sufficient knowledge and perception* of the system’s state, corresponding *safety strategies*...
and correctness of the application of the robot. While the correctness of application can be guaranteed (more or less) by the operator, the remaining elements have to be considered carefully for active safety measures. In this context they differ between measures describing safe situations focussing on single actuating components (e.g. via watchdog units and an emergency shutdown as also presented by [Tiruchinapalli2005]), and on the safe behavior of the robot. The authors use a modified situation operator model (SOM) in which each situation is described by a set of arbitrary data structures. Operators are then used for transitions between one situation to another and enable planning skills to the robot to reach a desired target situation. Each characteristic, as the data structures are called, also contains a risk value depending on the set of data values. In their definition risk is the product of accident severity and probability. For application, the qualitative expert knowledge has to be transferred to a quantitative description of the situations and their risks e.g. via fuzzy methods. An example illustrates the modeling of situations and risks in which a robot has the order to bring a plastic cup (C) to the kitchen. It has been modelled, that combining objects with the attribute plastic (cup) and heat (stove) is very risky. Therefore, a kind of risk map can be generated for the combination of objects with these two attributes depending on their distance as shown in figure 3.5. It is obvious, that the robot should not place the cup next to the kitchen stove (S). The given example only shows the functionality of the developed method. Further research will be focussed on better model descriptions to address more complex problems. This model description is the main challenge while it has to be precise and suitable to the given system and because of difficulties in formulating the dependencies of risks. Also expert knowledge will be incomplete and does not include all possibilities. Therefore, the authors offer the option that reinforcement learning will be used in future to adapt the risk assessment under supervision of a human operator. In the case of CRoMSCI there exists too few knowledge: Even an experienced operator is not able to predict whether the robot is able to handle a certain situation or whether it will fall down within the next second. Therefore, neither this kind of model knowledge nor a combination with reinforcement learning are suitable in the context of such climbing robots.

![Figure 3.5](image.png)

An interesting work – not regarding safety analysis, but because of corresponding measures in the range of climbing robots – has been published by Rachkov from the Russian Academy of Sciences. He discusses several safeguards in hard- and software, which are applicable for climbing machines [Rachkov1997]. Software measures can be summed up by supervision of system parameters, control of electronic components, and by routines to perform safe motion trajectories. He describes techniques like an emergency stop, which closes all vacuum grippers to preserve the current adhesion force or a track planning system moving on predefined safe trajectories. Hardware measures are additional technical
extensions like a safety cord or extra control cycles e.g. to shut off faulty elements. Besides negative pressure adhesion Rachkov also discusses safeguards related to magnetic or reactive force adhesion via a thruster. Other elements are emergency systems keeping the robot at the surface via rail guides, spikes, or pasting material or on-board devices like a brake parachute or damper material to reduce the effect of a drop-off. In fact, the author concentrates more on internal than on external hazards and introduces a couple of safeguards like a parachute, which are not suitable in the present case.

Nevertheless, it is also a common way in scientific robotics to implement safety measures without a previous analysis since there always exist hazards, which are obvious. A mobile system for example should always be able to evade or at least stop in front of an obstacle to avoid a collision. Until now a great number of sophisticated approaches has been introduced reaching from (direct) sensor-actuator-coupling [Castelnovi2005, Philippsen2003] up to learning systems, which collect and analyze data e.g. about traversability, roughness or other characteristics [Sun2005, Stavens2006]. Braun from the University of Kaiserslautern developed a cost-efficient navigation for a mobile outdoor robot, which includes a self-observation of the system as well as an estimation of the forthcoming terrain [Braun2008]. In his thesis the author uses different cost measures like the effort of a path (in terms of energy consumption) or the risk given e.g. via the distance between obstacles and the path. Of course, the robot has been equipped with underlying reactive elements keeping the system away from hazards. In this case large obstacles like trees and rocks, water expanses, or deep holes may harm the robot and must be avoided. Therefore, the robot has been equipped with multiple sensors like laser range sensors, bumpers, two stereo camera systems, a water detection device and further elements. By an evaluation of the sensor data and a merge with the vehicle data collected during the traversal the control system is able to estimate the further terrain characteristics. Based on this estimation, the vehicle can choose a path being less risky, faster or more economic – depending on the chosen cost function [Braun2009]. Unfortunately, this and other promising approaches are not suitable in the present case since there exists no sensor system, which can be applied to detect necessary environmental features in front of a climbing robot (compare section 2.1.5).

3.3.3 Discussion of Safety Approaches

To sum up the state-of-the-art it can be recapped that there exist many different ways of analyzing, handling, and improving robot safety. While some researchers concentrate on internal components and their malfunctions, others try to consider the interaction between robot and environment. In the present case those approaches using statistical data of failure rates seem to be not helpful since the acquisition of these data is not possible. In the same way model-based methods as introduced by Ertle et al. are interesting, but not applicable since there exists no sufficient description of the interaction between robot and environment. Nevertheless, analysis techniques like fault tree analysis seem to be suitable tools to identify hazards and their causes and to ascertain possible safety measures to reduce the risk caused by them. The next section will introduce hazards for a climbing robot, which have to be considered to get a safe system.

5Because of a high weight of the climbing robot a parachute must have an enormous size and needs a certain drop height to unfold, which makes it inapplicable.
3.4 Hazards for Climbing Robots

As shown in the previous section, a couple of guidelines and methods exist to develop a safe robotic system or to analyze it regarding safety aspects. The first steps are the determination of system parameters, limits, and the identification of hazards. Climbing robots like CROMSCI face additional hazards compared to ground vehicles and have to use other or enhanced safety rules. Again, human safety will not be explicitly considered here while the main hazard for the technical staff is linked to the worst case scenario of the robot: The loss of adhesion and a drop-off. Table 3.3 gives an overview on potential hazards for climbing robots using negative pressure adhesion, a drive system and inflatable sealings. Based on this table, an analysis of the hazards, their consequences, and possible causes has to be done. The risk value of each of these hazards is hard to determine because the probability of occurrence strongly depends on dynamic effects during operation and on the environment. In this way the work of [Rachkov1997] is dissatisfactory because most of his methods are “after the fact”-measures. Of course, the robot does not fall down if it is secured by a rope. But, this situation should also be avoided since even a secured drop-off may lead to strong damages at the robot chassis, the manipulator arm or at the expensive inspection sensors. For best performance the rope has to be equipped with a locking retractor as used in seat belts to allow maximum flexibility during operation, but a sudden stop in case of a drop-off. To reduce sideward swinging to a minimum a second attach point at the building is needed. Finally, the rope has to create a triangle between retractor, robot and the second mounting point. Nevertheless, even a secured drop-off remains dangerous and should be avoided under all circumstances. This counts at least if economic aspects become important and the robot has to be rescued manually each time what increases the costs for inspections due to less time in operation.

As summarized in table 3.3, there exist various hazards for climbing robots, which may lead to a drop-off or severe damages. Some of them are based on internal errors like power or connection failures, malfunctions of electronic components or software bugs. Others have a dynamical nature and can only occur while the robot interacts with its environment. The table also shows that it is not easy to determine one class of severity for a hazard because this also depends on the robot state, its current situation and dynamic effects. As an example, the degree of severity of a hole in the rubber sealing caused by penetration strongly depends on its size. If the hole is very small the pressure controller might still be able to balance out the loss of air. In contrast to that the whole sealing might be unusable if the hole is too big. In this case the robot will fall down since the system is not able to air-proof the adhesion chambers. There also exist hazards linked to high mortality parts increasing over time like the abrasion of the wheels. Because of their mechanical structure climbing robots like CROMSCI have some elements, which have to be maintained periodically. The drive e. g. always produces some wheel slip, thus the wheel rubber will wear off. The risk of a broken wheel will increase depending on the distance the robot drives, the inclination angle of the wall, the distribution of the downforces on the wheels, the surface’s roughness and additional parameters. Of course, the wheel itself will not be broken, but the rubber can be damaged in a way that it will be destroyed. Then robot motion will no longer be possible and the robot has to be rescued manually. In this case the severity class will change during runtime.

The next sections will introduce internal and external hazards and possible risk reduction and avoidance techniques.
Table 3.3: Potential hazards for climbing robots using negative pressure adhesion, inflatable sealings and a drive system.

<table>
<thead>
<tr>
<th>hazard / event</th>
<th>severity class&lt;sup&gt;a&lt;/sup&gt;</th>
<th>immediate consequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>tilt</td>
<td>1</td>
<td>robot crash (drop-off)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>damages at wheels</td>
</tr>
<tr>
<td>slip</td>
<td>1</td>
<td>robot crash (drop-off)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>damages at wheels or sealings, collision or robot stuck</td>
</tr>
<tr>
<td>collision</td>
<td>2</td>
<td>severe damages at chassis or sensors</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>damages at chassis</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>damages at manipulator</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>damages at inspection sensors</td>
</tr>
<tr>
<td>driving into hole</td>
<td>1</td>
<td>severe sealing damages (drop-off)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>severe wheel or robot stuck</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>damages at wheels</td>
</tr>
<tr>
<td>wheel abrasion&lt;sup&gt;b&lt;/sup&gt;</td>
<td>2</td>
<td>severe damages at wheels</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>damages at wheels</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>reduced grip</td>
</tr>
<tr>
<td>leaky sealing</td>
<td>1</td>
<td>robot crash (drop-off)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>falling out of one working chamber&lt;sup&gt;c&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>reduced leak-tightness of chamber</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>adaptations by pressure controller</td>
</tr>
<tr>
<td>internal malfunction&lt;sup&gt;d&lt;/sup&gt;</td>
<td>1</td>
<td>robot crash (drop-off)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>fail of safety measures or locomotion</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>fail of navigation strategies</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>fail of manipulator</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>loss of inspection data</td>
</tr>
</tbody>
</table>

<sup>a</sup> The corresponding severity classes for each hazard is given according to table 3.1. The probabilities and hence the resulting risks are not considered here.

<sup>b</sup> Wheel abrasion is a continuous problem. The risk of a failure increases during runtime depending on the travelled distance and other criteria.

<sup>c</sup> Depending on the setup of the chamber sealings. In the current version of CROMSCI with one global sealing, all working chambers would be affected by a damage.

<sup>d</sup> Internal malfunctions include single, multiple, or global failures (black-out, wrong data, etc.) of electronical components like sensors, actuators or circuit boards, communication errors between the individual electronic components, power failures, controller and software faults, and other malfunctions with regard to the embedded control system.
3.4. Hazards for Climbing Robots

3.4.1 Internal Hazards

Internal hazards occur because of malfunctions of the system itself without any influence from outside. Beside the robot itself also its supply and control station has to be considered here if the system is not fully energy autark and autonomous. In general, internal hazards include circuit defects of the electronics, malfunctions of sensors, motors, communication devices, power supply or defects of the mechanical structure. In general, a component does not have to fail completely to cause an incident. A malfunction can also include e.g. time delays between distributed devices because of a high bus load, which disturbs communication and further processing. These hazards can be analyzed with existing methods – although they are still a big issue in research – to find and apply suitable safety measures (e.g. watchdog units or redundant components). Many manufacturers in industry specify failure rates and probabilities like MTTF for their products and guarantee a certain life span. Others analyze the software components or embedded controllers via stress tests or verification techniques to detect possible errors. Especially the verification of software is a growing field and becomes more and more complex in the range of distributed (embedded) systems with a large amount of components. A robot e.g. is equipped with plenty circuit boards and electronical elements, which communicate via a bus system or direct link with control computers, sensor nodes or actuators. It is a tremendous problem to proof that these systems work correct since there exist myriads of error sources in hardware, software, or communication as examined e.g. by [Zimmermann2009, Hussain2010]. In the range of climbing robots, of course the malfunction of a hardware component (e.g. a chamber valve or a pressure sensor) or the control software (e.g. a chamber pressure controller) cannot remain unconsidered, since the severity of these hazards reaches up to a total loss of the robot. Nevertheless, it seems to be legal to ignore these internal faults since the external hazards, which will be described in the next paragraphs occur much more often as also shown later in section 5.1. The main assumption is that the probability of internal errors is much smaller than the one of external errors.

Nevertheless, internal hazards have to be taken into regard in the development and design phases of a climbing robot. If an internal error occurs additional safety measures have to be implemented to avoid this incident again. Here methods like cause and effect diagrams or a fault tree analysis are very useful to detect the causes. Afterward the development and implementation of the best reduction mechanism is straight-forward in many cases.

In addition to the definitions given at the beginning of this chapter, risks can be influenced by safeguards, which lower the probability of a hazard to occur. Kaplan and Garrick formulated that risk is a hazard divided by corresponding safeguards [Kaplan1981]. This symbolically equation is not hundred percent consistent to the definition used here since the probability of a hazard to occur is missing. But, as it can be seen it is possible to lower the internal risks (the probability of occurrence of harm) by adding safety measures. cromsci can be taken as an example here: During the first experiments on a concrete wall in 2008 the secured robot lost its adhesion several times while it was driving. The cause for that was a problem with the external power supply. Every time the wheels were steered the power consumption of the motors increased so much that the pressure valves, which are responsible for the air pressure inside of the sealing tube, were shut off. As a result, the sealing was no longer supplied with air and all chambers lost their
Table 3.4: A classification of the consequences and final safety measures, which can be applied if that incident occurred.

<table>
<thead>
<tr>
<th>consequence class</th>
<th>effect on operations</th>
<th>final safety measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (most severe)</td>
<td>robot crash^a</td>
<td>none / security rope</td>
</tr>
<tr>
<td>2</td>
<td>robot inoperable</td>
<td>manual rescue via security rope</td>
</tr>
<tr>
<td>3</td>
<td>robot adhesion or navigation affected</td>
<td>adaption of safety strategies, driving to home position</td>
</tr>
<tr>
<td>4</td>
<td>inspection inoperable</td>
<td>abortion of inspection task, driving to home position</td>
</tr>
<tr>
<td>5 (least severe)</td>
<td>no or neglectable effect</td>
<td>default</td>
</tr>
</tbody>
</table>

^a In fact, a robot drop-off will be avoided by a security rope. But, of course also the mounting or the rope itself may fail. In every case, the consequence class of this incident can only be lowered to level 2, because the robot might swing and hit the wall.

leak-tightness. By adding a second individual power supply only for the drive system this problem could be fixed and the risk of this failure could be minimized.

In general, it is possible to handle most of these internal hazards by redundant or additional components as shown in the example. Nevertheless, the usage of these safeguards is limited due to their weight and the available space. It is generally accepted, that these safety measures are suitable for mobile robots, but they are counterproductive for climbing robots if e. g. the weight increases too much. In this case it would be good if CROMSCI had more individual working chambers with a separate suction engine and sealing system to be protected against single failures. This would lower the risk of the internal and some external hazards, but would increase the required downforce and the wear of the wheels. For robot developers it is very hard to compare both influences and to find the optimum concerning safety and complexity of the system.

3.4.2 Collisions

In contrast to the internal risks the external effects are varying a lot: A couple of them can be described and handled well, some increase over time, others again are hardly to characterize. Additionally, some of the hazards given in table 3.3 can cause other events. For instance, it is possible that the robot slips and thus collides with an obstacle, which damages the manipulator. The final safety measures – if all other measures fail – are shown in table 3.4.

Collisions are a typical threat for mobile robots and their prevention is a common research area since decades. In the case of large vertical walls mainly salient and overlapping structures may harm the robot chassis. Collisions can be associated with a severity level of 2 since they may cause an inoperable system, as listed in table 3.4, but no drop-off. In contrast to standard ground vehicles dynamic objects can be neglected because it is not expected that other climbing robots or industrial climbers act in the same workspace. There exist different parameters, which determine whether an obstacle – in this context one speaks of macro structures – can be overcome by the robot or not. As illustrated in


figure 3.6, these parameters are obstacle ground distance \( h_{O}^{pos} \), height of an overhanging structure \( h_{O}^{badge} \) or a depth \( h_{O}^{neg} \), the ground clearance of the robot \( h_{R}^{clear} \), its chassis height \( h_{R} \), the indentation of the sealing \( h_{S} \) and the wheel radius \( r_{W} \). For the given setup two classes of impassable positive obstacles can be distinguished:

- **Protrude obstacles**: Positive obstacles like salient structures are passable if their height \( h_{O}^{pos} \) does not exceed the maximal sealing indentation \( h_{S}^{max} \). Therefore, three states can be defined: A protrude obstacle is *passable* if \( h_{O}^{pos} < h_{S}^{max} \), it may *endanger the sealing* if it is lower than the chassis ground clearance \( h_{R}^{clear} \), but greater than the maximal sealing indentation \( h_{S}^{max} \leq h_{O}^{pos} < h_{R}^{clear} \), or it *endangers the chassis*, if \( h_{O}^{pos} \geq h_{R}^{clear} \). In the present case, the maximal sealing indentation lies at about 6 mm which thus is the upper limit of positive obstacles (although the sealing might be damaged by these objects, which is considered later).

- **Overlapping structures**: In a similar way overlapping structures as shown in figure 3.6 have to be handled. The robot can pass them if its chassis height \( h_{R}^{chassis} \) is below the obstacle's distance from the ground: \( h_{R} < h_{O}^{badge} \). Otherwise the chassis collides with the obstacle. In case of CROMSCI this would happen if the structure is lower than 0.4 m from the ground.

It is expected that in both cases the chassis might be damaged. The severity strongly depends on the strength of the impact as well as on the position of collision. In the worst case it is expected that e.g. the manipulator arm is broken in a way that the locomotion of the robot cannot be continued. General safety measures concerning collisions include both risk avoidance and risk reduction. Collision avoidance is attended by sensor processing, obstacle detection and locomotive action. It strongly depends on the expected environmental features, the robot setup and additional parameters. At first, a maximum height for passable obstacles has to be defined. Then, an appropriate sensor system has to be selected, which provides the desired range, accuracy, and speed to reliably detect these features. Based on these data, the robot can apply safety measures, which let it evade the detected obstacles by turning away or by slowing down the system to diminish the effect of an impact, as presented later in chapter 9.

---

\(^{6}\)In fact, the sealing indentation is not the only limiting factor in this case. Due to the wheel setup (compare section 2.1.1) the maximal height, which can be overcome by a wheel is 6 mm. Therefore, the maximal sealing indentation is set to this value, although it could be larger in general.
3.4.3 Driving into a Hole

Beside the wide range of different obstacles there also exists myriad shapes of holes, gaps, or deepenings endangering the robot. This can be desired grooves at connection points of different building parts, but also unwanted defects resulting from weather influences or structural damages (compare figure 1.3). A driving into an impassable hole could lead to severe damages at the wheels or the sealings. The effect reaches up to a drop-off of the robot if the sealing is penetrated and destroyed.

- **Structure edges:** Figure 3.7 shows two situations at the end of a concrete structure or at a large step. On the left side (figure 3.7a) at least one chamber reaches over the end of the structure and although the sealing \( S_{1o} \) is extended to its maximum the gap obviously cannot be sealed. This would result here in a loss of adhesion force and a moment \( M_{R|y} \) around the robot’s y-axis, which could cause the robot to tilt if this could not be compensated by the remaining working chambers (more information about this case can be found in section 3.4.7). In figure 3.7b the robot managed it to move on until the frontal wheel \( (W_1) \) passes over the edge and loses contact. In this case the robot is tilted with angle \( \theta_R \) and may gets stuck because the chassis rests on the remaining wheels and the sealing. Additionally, the sealings may be damaged because they are rubbed between wall and chassis.

![Figure 3.7: Large negative obstacles are possible causes for robot tilt because of leak chambers (a) or that it gets stuck and damages the sealings (b).](image)

- **Single deepenings:** Beside large structure edges also single holes or gaps present a serious danger. At first, they may damage the wheel rubber because of sharp edges. Otherwise there is the chance that the robot gets stuck if the hole is too large. Figure 3.8 shows the relationship between wheel radius \( r_W \), sealing indentation \( h_S \) and hole length \( d_{O|x} \) in rolling direction of the wheel.

Three conditions must be satisfied to let the wheel cave into a hole and let the robot stuck: First, the width \( d_{O|y} \) of the hole must be larger than the wheel width \( d_{W|y} \). Second, the hole must be deeper than the maximal sealing indentation \( h_S^{max} \). Finally, the hole has to be longer than the maximum length \( d_{O|x}^{max} \) as given in equation (3.2).

\[
d_{O|x}^{max} = \sin \left( \arccos \left( \frac{r_W - h_S^{max}}{r_W} \right) \right) \cdot r_W \cdot 2\]

(3.2)
Therefore, a negative obstacle \( O \) is impassable for sure if the following equation (3.3) with the hole dimensions \( d_{O|x} \), \( d_{O|y} \) and \( d_{O|z} \) is satisfied. Additionally, the grip at the contact point between wall and wheel also influences the trafficability of such a negative obstacle.

\[
\text{impassable}(O) = d_{O|x} > d_{O|x}^{\max} \wedge d_{O|y} > d_{W|y} \wedge d_{O|z} > h_S^{\max} \quad (3.3)
\]

These hazards are handled in a similar way as collisions with obstacles: The surface has to be detected with appropriate sensors with the required resolution and the robot has to evade these negative obstacles if they are too dangerous or impossible to pass. Also the reduction of velocity up to a full stop is useful if the system faces such a hazard to keep a certain distance from it.

3.4.4 General Robot Roll and Tilt

The trafficability of obstacles and holes also depends on the current wheel positions. If one or more wheels are located above or below the normal ground plane the robot is rolled and tilted, which influences the sealing indentation. This becomes obvious in figure 3.9 showing the frontal wheel located inside adhesion chamber \( C_1 \) positioned on a small positive step. This lets the robot tilt with angle \( \theta_R \). Because of this the frontal sealing \( S_{1o} \) is expanded while the rear sealing \( S_{4o} \) is impressed – the dimensions of the maximum passable holes or obstacles shrink. In the worst case, this effect can cause damages at the sealings in the same way as in the case of driving into a hole (robot drop-off, severity level 1) since the
sealings might be crushed between chassis and surface. But, it is expected that the robot will first get stuck if the sealing is impressed to the maximum and the sealing material is not worn off.

Figure 3.9: The influence of robot tilt on the opposite sealings \( S_{1o} \) and \( S_{4o} \). The one side is impressed (right) while the other (left) is expanded, which has an effect on trafficability.

Because of the possible impact on the robot, the causes and situations in which these problems occur have to be figured out more detailed, as shown in appendix A.4.2. The calculated expansion and indentation values of the sealing are not necessarily the maximum values during the complete phase of handling a step or an obstacle. It should be obvious, that impression and expansion have at least the same amount as the obstacle height or depth in some situations (e.g. if only the sealing is over the obstacle while all three wheels are still on the ground surface). Based on the calculation steps given in appendix A.4.2, it can be decided if an obstacle – which is passable by the drive system – can be handled by the sealing system without running the risk of getting stucked (needed sealing indentation too high) or of loosing one or more working chambers (needed sealing expansion too high). In general, the height limits of passable holes and obstacles are lowered to ensure robot safety.

The importance of these examinations about the effect of roll and tilt can be visualized via an example. If the frontal wheel overcomes a step of height 0.5 cm as shown in figure 3.9, the situation can be described as if the rear wheels are inside of a hole with \( h_{O_2} = h_{O_2} = -0.005 \text{ m} \), the lower height offset will – because rear wheels and rear sealing are on the same level – also be \( h_{S_{\text{low}}} = -0.005 \text{ m} \), whereas the higher offset is \( h_{S_{\text{high}}} = 0 \) (compare equations (A.21) to (A.24)). With the robot setup (sealing radius \( r_S = 0.36 \text{ m} \), wheel distance \( l_W = 0.26 \text{ m} \)) the tilt angles of the robot are \( \theta_R = -0.735^\circ \) and \( \phi_R = 0^\circ \), the spherical angles describing the total inclination of the wheel’s plane are \( \gamma_n = 0.735^\circ \) and \( \delta_n = 180^\circ \) (compare figure A.3). Finally, the needed sealing expansion is about \( h_{S_{\text{exp}}} = 1.28 \text{ mm} \) at the front according to equation (A.32) whereas it is impressed about \( h_{S_{\text{ind}}} = -2.96 \text{ mm} \) at the rear of the robot (equation (A.31)). This means that the maximum indentation reached half of the sealing’s maximum. If the ground at the rear side around the sealing would be on the same level as on the robot front (so only the rear wheels are located inside of a hole or a ground depression of 1 cm depth) also the lower sealing offset \( h_{S_{\text{low}}} \) would be zero and the needed pressing would be \( h_{S_{\text{ind}}} = -7.96 \text{ mm} \), which is larger than the hole’s depth and a more than the system can manage.
3.4.5 Wheel Abrasion and Wheel Slip

Wear of tires is a general issue for wheel-driven vehicles. This effect of abrasion results from wheel slip, which occurs if a wheel has to transmit forces between vehicle and ground. Generally it can be differed between a tangential force $F_{W|x}$ in rolling direction of a wheel – because of acceleration and deceleration – and a lateral force $F_{W|y}$ because of sideward forces (e.g. steering). In the case of wheel-driven climbing robots, gravity is a main problem for the drive system. The drive motors have to generate high forces to overcome the weight of the robot and to push it up the wall. These forces are transferred via the tires interacting with the surface in two ways: The first effect is sticking between rubber and surface, the second is the effect of toothing between both materials depending on the roughness values. If the wheel slips it should be obvious that – because of that sticking and toothing – some of the rubber material will be ripped out and remain at the surface. This abrasion could lead to wheel damages up to a broken wheel rubber, which makes robot motion impossible. In the worst case, the robot will stuck at the current position and has to be rescued by hand (severity class 2). It is not expected that the robot can drop off because of this general wheel slip. Nevertheless, it has to be taken into account because a multiple wheel slip can cause a dangerous slip of the complete robot as it will be described in section 3.4.8.

The coefficient of static friction $\mu_{stat}$ results from both effects and describes the amount of transferable forces longitudinal to the ground plane. This amount corresponds to the maximal holding force $F_{W|xy}^{\max}$, which has to be overcome to pull an object over the ground. It can be calculated as given in equation (3.4) depending on the normal force $F_{W|z}$. The sum $F_{W|xy}$ of lateral and tangential forces at the wheel must not be larger than this maximum to avoid slip (see friction circle in figure 3.10 and equation (3.5)) [Pacejka2005]. The static friction coefficient\textsuperscript{7} of rubber (R) and dry concrete (C) is at about $\mu_{stat,RC} \approx 1.0$ whereas the kinetic (or sliding) friction value $\mu_{kin,RC}$ lies between 0.6 and 0.85. The problem of wheel abrasion is strongly connected to the shown friction coefficient. It is obvious, that the abrasion will be minimized if the friction value is close to zero – unfortunately a high value is needed to transfer enough propulsion force for robot motion.

\[ \mu_{stat} \cdot F_{W|z} = F_{W|xy}^{\max} \]  
\[ F_{W|xy}^{\max} \geq F_{W|xy} = \sqrt{(F_{W|x})^2 + (F_{W|y})^2} \]  

Another aspect is the wheel slip $S_W$. Wheel slippage is the ratio of the wheel’s rotational velocity $\omega_W$ to the theoretical rotational velocity of a passive wheel $\omega_0$ and is additionally affected by the transferred forces. It occurs, if forces have to be transmitted between wheel and surface – hence also for acceleration and for deceleration. The basis for the

\textsuperscript{7}These friction coefficients vary over the different sources and should give only a rough idea of common values, e.g. from http://de.wikipedia.org/wiki/Reibungskoeffizient and http://en.wikipedia.org/wiki/Coefficient_of_friction
force transfer are complex effects at the contact patch. At this area the rubber is deformed and performs local sliding movements [Reif2010]. Equation (3.6) describes how the slip value can be calculated. Equivalent to the given description the slippage can also be seen as the ration of robot velocity $v_R$ to the velocity of the wheel (with wheel radius $r_W$). This value tends to infinity if the wheel is turning without moving the robot ($\omega_0 = 0$ resp. $v_R = 0$) and will be zero if all wheel turning is used for robot motion ($\omega_0 = \omega_W$). In case of a full braking one speaks of wheel locking if its rotational velocity is zero while the vehicle is still in motion, the slip will be $S_W = -1$. An interesting aspect is that the friction coefficient reaches a maximum at a specific amount of wheel slippage. In the case of a car, this value lies at wheel slippage of $10\%-20\%$ for dry asphalt.

$$S_W = \frac{\omega_W - \omega_0}{\omega_0} = \frac{\omega_W \cdot r_W - v_R}{v_R}$$ (3.6)

In general, the total driving resistance $F_{D|\text{resist}}$, which has to be overcome by the drive system, is the sum of rolling $F_R$, air $F_A$ and climbing resistance $F_C$. The difference of the applied propulsion force and this total resistance force accelerates the vehicle. In the case of a climbing robot like CROMSCI, the air resistance force can be neglected because of the low velocity ($F_A \approx 0$). Unfortunately, another force – which does not exist in automobiles – has to be considered here: The resistance of the sealings $F_S$. Equation (3.7) shows the components of the total driving resistance with the rolling resistance coefficient $c_R$ having a value of 0.01 to 0.015 (for a rubber wheel of an automobile on concrete ground\textsuperscript{8}) the downforce $F_{D|z} = \sum F_{W|z}$ of the drive, vehicle weight $F_{R|g}$, slope angle $\alpha$, friction coefficient of the sealing $\mu_S$ and normal force at the sealing $F_{S|z}$. Figure 3.11 illustrates the different forces.

\textsuperscript{8}http://en.wikipedia.org/wiki/Rolling_resistance
3.4. Hazards for Climbing Robots

\[
F_{D|\text{resist}} = c_R \cdot F_{D|z} + F_A + F_{Rg} \cdot \sin \alpha + \mu_S \cdot F_{S|z}
\]  

(3.7)

Unfortunately, the wheel slip in rolling direction is not the only problem: Another source of slippage results from shear forces lateral to the wheel, as shown in figure 3.12. These unmeant forces can occur because of steering transitions, incorrect initializing and runtime errors\(^9\) of the turning wheel domes. They produce a deformation of the wheel rubber in lateral direction, which can cause abrasion, but also increase the probability of wheel slip because the maximal transferable force in rolling direction \(F_{W|x}^{\max}\) of a wheel is lowered by sideward forces as given in equation (3.8). Therefore, an additional control system of the wheel steering is needed to minimize contrarily orientations for reducing sideward forces.

\[
F_{W|x}^{\max} = \sqrt{(F_{W|xy}^{\max})^2 - (F_{W|y})^2}
\]  

(3.8)

Figure 3.12: Problem of shear forces: The maximal transferable force in rolling direction of each wheel is lowered. In this case wheel \(W_2\) will slip because the wheel force \(F_{W_2|xy}\) exceeds the friction circle.

Solutions for wheel abrasion and wheel slip are special avoidance techniques like a traction control system (TCS), a shear force controller (SFC) or rules, which limit certain motions. Depending on statistical values and on an estimated level of abrasion the wheels should be checked periodically. This could be triggered by the robot itself by driving down the wall automatically and requesting an inspection of the wheel rubber.

\(^9\)Runtime errors can be caused by different sliding resistances of the rotating wheel domes resulting in different steering speeds, if the closed-loop controllers cannot balance out this conflictive force.
3.4.6 Leaky Sealing

Beside the rubber wheels also the inflatable sealing is endangered of getting damaged. Two main threats exist here: Sharp edges and abrasion. Sharp edges exist because of defects and inaccuracies during construction of the concrete building by using inaccurate sheathing plates. These protrude concrete pikes or crests can cut the side of the sealing. The resulting hole can be so large that the sealings controller will not be able to regulate the pressure anymore. As a result, the sealing will lose its leak-tightness and the robot might drop off (severity level 1). Another problem arises from the abrasion of the sealing’s sliding coat since also the air-proof sealing material might be damaged. The chance of both threats grows in case of general robot tilt (compare section 3.4.4) or a driving into a hole (section 3.4.3), which might increase the friction force as well as the chance of a crushing of the sealing.

In both cases there exists no single measure to make the system safe. Of course, the system should detect and evade sharp edges automatically by evasion strategies. The problem in this case is of course the detection of these very thin structures. Another measure is the sealing controller including the pressure generator, which should be able to handle smaller leakages by having sufficient pressure reserves. Finally the material itself has to be optimized to be as robust, flexible and sliding as possible.

3.4.7 Robot Tilt

As given in table 3.3, robot slip and tilt are very serious problems concerning the adhesion. In his thesis Hillenbrand points out, that both effects have to be avoided to ensure the safety of the system [Hillenbrand2009]. Figure 3.13a illustrates the consequences of robot tilt. The lower sealing rests on the surface, the front part of the robot is lifted including the wheel and the upper sealing elements. Thus, too many chambers lose their negative pressure and the adhesion system will not be able to produce enough downforce to hold the robot.

Tilt itself results from gravity affecting the robot at its mass center, which is located \( h_{R,m} \) above the robot center \( R_C \), as shown in figure 3.13b. To counter this torque only the drive system will be used – all forces fo through the three wheels spanning a stability triangle (see figure 3.14). For simplicity the influence of the sealing, which also absorbs some torque forces, will be neglected. By taking the inclination angle of robot environment \( \alpha_E \) into account, the total mass torque \( M \) can be calculated according to equation (3.9).

The torque increases linear to the distance of the mass \( h_{R,m} \), which makes it necessary to lower the center of mass as much as possible. The most difficult setup for CROMSCI is an inclination angle of about \( \alpha_E = 120^\circ \) and a robot yaw of \( \psi_R = 0^\circ \) because of a high torque and a short lever arm.

\[
M = m_R \cdot g \cdot h_{R,m} \cdot \left( \frac{1}{\tan \alpha_E} \cdot \cos \alpha_E + \sin \alpha_E \right)
\]

As already mentioned, it is assumed that all forces and torques go through the wheels. This includes the tilt torque, but also the downforces generated by the negative pressure chambers. The stability property of non-tilting can be formulated as the situation in
which all wheels are attached and pressured to the wall. If one wheel loses contact it will theoretically receive a negative downforce and the robot will tilt. Therefore, equation (3.10) must be satisfied to avoid robot tilt with the downforce values $F_{W_i|z}$ at all three wheels.

$$F_{W_i|z} \geq 0 \quad \forall i \in \{1, 2, 3\}$$

This equation corresponds to figure 3.14 in the way, that the center point of drive’s downforce $(x_{D|F_z}, y_{D|F_z})^T$ – given in the robot coordinate frame – has to lie within the triangle constructed by the three wheel contact points. Based on the geometric parameters of the three wheels, this requirement can be achieved by satisfying the inequalities of equation (3.11). Of course, the total downforce of the drive $F_{D|z} = F_{W_1|z} + F_{W_2|z} + F_{W_3|z}$ is positive if every single wheel force is positive according to equation (3.10).

$$-\frac{1}{\sqrt{3}} \cdot x_{D|F_z} + \frac{1}{\sqrt{3}} \cdot l_W \geq x_{D|F_z} \geq -\frac{1}{\sqrt{3}} \cdot l_W$$

The robot tilt cannot be avoided and has to be balanced out by the adhesion system and the drive. The situation shown in figure 3.14a creates high overturning because of the short lever arm. In contrast to that the robot is able to absorb higher overturning if one wheel is located below the center of mass (figure 3.14b). Unfortunately, the system has to be well balanced in this situation otherwise it would tilt over the sides. Therefore, the area of stability can be approximated by a circle with radius $\frac{4}{3}l_W$, which fits inside the triangle and is independent from the robot orientation. A general condition for stability can be described as shown in equation (3.12):
The point of downforce has to lie within the wheel triangle (a). Depending on the robot orientation $\psi_R$ the maximum possible tilt varies (b). If the center point lies outside of this triangle one wheel loses contact and the robot tilts (c).

\[
\sqrt{x_{D/F_z}^2 + y_{D/F_z}^2} \leq \frac{l_W}{2}
\]  

The best way to handle the tilt of the robot is to change its geometry. One solution is to decrease the center of mass to reduce the mass torque. A second solution is the enlargement of the stability triangle by moving the wheels to the outside of the negative pressure system. In both extreme setups – center of mass located on the wall plane or all negative pressure chambers inside of the wheel triangle – the problem of tilt does no longer exists. Unfortunately, both ways are not suitable or possible because of the mechanical construction, limitations in robot size and further restrictions. Therefore, tilt has to be handled in another way. Because of the setup of the seven adhesion chambers it is possible to control the location of the downforce point. To compensate robot tilt the chamber pressures have to be adapted and distributed in a special way: The upper chambers have to produce higher adhesion forced while the lower ones have to release a bit. The goal is to balance the downforces at all three wheels so that the forces are distributed equally among the drive system. As a side effect, this will also prevent an irregular abrasion of single wheels. Unfortunately, the problem of robot tilt is increased by the fact that not all negative pressure chambers can reach their desired pressure value. Reasons for that are leakages at the chamber sealings due to gaps or micro structures whose effects on the adhesion system are hardly predictable and almost impossible to model. In the scope of this thesis, some necessary adaptions and important improvements are presented to make the system more robust against external influences.

### 3.4.8 Robot Slip

The other most serious problem is robot slip. In contrast to wheel slip as described in section 3.4.5 does robot slip denote the gliding of the whole robot. The focus lies here on undesired motion of the robot itself in direction of gravity, which can cause the robot to drop off since it may slip towards a deep defect causing high leakages. This corresponds to a maximum severity level 1. Robot slip is not necessarily linked to turning wheels, which
do not have enough grip. It can also occur if the drive system tries to hold the robot at its current position. Similar to wheel slip it depends on the downforce at the wheel contact points—hence on the adhesion forces generated by the adhesion system—and on the friction coefficients between wheels and surface. Therefore, the attribute of non-slipping can be ensured only by the three wheels, which need enough grip and adhesion force supported by the negative pressure system to hold the robot weight. Again, the effect of the sealing on robot slip will be neglected because its friction should be minimized for low counteracting forces during robot motion. To calculate the forces affecting the robot drive a common friction value $\mu_{\text{stat}}$ for all wheels and no torque around the robot $z$-axes are assumed. Now equation (3.5) can be transformed to be valid for the whole undercarriage of CROMSCI (equation (3.13)) according to the friction circle as already shown in figure 3.10:

$$\mu_{\text{stat}} \cdot F_{D|z} = F_{D|xy}^{\max} \geq F_{D|xy} = \sqrt{(F_{D|x})^2 + (F_{D|y})^2}$$

(3.13)

Therefore, the constructional parameters to reduce robot slip are a high friction value $\mu_{\text{stat}}$, a high downforce $F_{D|z}$, and a lower climbing resistance $F_C$, which can be adapted e.g. by reducing the robot weight (see figure 3.11). Nevertheless, slippage remains a serious problem because the robot might slide down and reach the end of the structure, a deep gap or obstacles. In some cases a small portion of robot slip cannot be avoided, e.g. because of wheel rubber deformations if the robot horizontal on the vertical structure. In this case the robot will slightly lose height, which can only be avoided by an adaption of the wheel steering so that the robot tries to drive parallel and a bit upwards. Unfortunatley, this and other dynamic effects caused by a porous or uneven surface and the vehicle dynamics are hard to model and thus nearly impossible to predict. Additional slip reduction techniques are advanced motion controllers and an adhesion controller.

### 3.4.9 Discussion of Hazards

It becomes obvious, that a climbing robot is more or less always endangered and that there is always the risk of a drop-off. Of course, it is also possible that combinations of these hazards occur or that e.g. a malfunction inside of a controller in combination with the current surface patch causes a drop-off, although seperately none of those hazards would be a serious problem. Some of these hazards are also correlated as described before. Of course, small steps, which can be overcome by the robot, do not only produce robot tilt. Additionally, the chance of wheel abrasion or damages at the sealings increases, which either may cause problems at once or raise the chance of a malfunction during further operation.

As presented before, most of the hazards could lead to a drop-off of the system. This corresponds to the highest severity level since not only the robot, but also persons are endangered. In fact, it is not possible to eliminate these hazards completely. There always exists the chance that a component fails, including the security rope as final safety measure or the mounting at the robot’s chassis. But, it is possible to reduce the likelihood of their occurance and thus reduce the risk. At that point it should be obvious that an examination of possible causes for these hazards is mandatory to find suitable measures
to decrease their probability of occurrence. The next chapter will present the challenges concerning safety aspects of mobile climbing systems and necessary aspects of this thesis concerning risks, safety, and the problems in creating the required models for classic analysis techniques.
4. Thesis

As shown so far, there is a large gap between standard risk assessment methods and a complex dynamic system. Whereas some methods need statistical data about the probabilities [Tiruchinapalli2005, Robinson1996] or model descriptions [Wardzinski2008, Ertle2010] to provide risk information, other methods are only able to detect sources of risks [Jordan1972, Mears1995] and to check if certain safety requirements are fulfilled [Gulland2004]. The applications of the risk assessment and handling methods in literature are in general limited to well-known situations and robotic systems whose interaction with the environment can be described well.

The main question linked to safety of climbing robots is: “When is the climbing robot going to fail and how can this be prevented?” Although the robot has to be secured by wire as it is demanded by law even a secured drop-off might damage the robot chassis, the manipulator or its sensors. In the best case, the inspection is interrupted and the robot has to be rescued by hand. Furthermore, even complex closed-loop control structures as developed by [Hillenbrand2009] are not sufficient for safe navigation of such a climbing robot, since there always exist environmental conditions, which cannot be handled by the adhesion and locomotion system. Therefore, analysis techniques and safety measures are required to reduce the operational risk during navigation to an – at least – tolerable level, as demanded in thesis 1.

**Thesis 1**

Safe navigation of a wall-climbing robot can only be achieved by methods of risk estimation and prediction to detect dangerous situations and by appropriate safety measures to reduce or avoid risks.

The challenges for climbing robots regarding safety analysis and handling thus arise from several limitations. Most of these limitations are linked to the more or less unique setup and application of this technical system.
Interaction model The interaction between robot and environment cannot be described sufficiently. A thermodynamical model for simulation of the airflow has been set up by Wettach et al. [Wettach2005a] and enhanced in further steps to get more realistic results. Also the sealings tightness is simulated with a simple model. Nevertheless, the complete navigation and adhesion system consisting of wheels, sealings and negative pressure chambers, and its interaction with an a priori unknown surface cannot be modelled satisfyingly. Wheel abrasion, sealing friction resistance, and leak tightness depend on surface parameters like friction coefficient and roughness, which are hardly to determine.

Failure data The interpretation of failures is very difficult due to missing model knowledge. First of all, it is not easy to determine the sources of a failure. Untight chambers e.g. can be caused by grooves or cracks, surface roughness, air channels, but also by missing sealing pressure or robot tilt. Especially the characterization of the surface is very hard because of unavailable corresponding sensors. Furthermore, the current demonstrator CROMSCI is not reliable enough to collect sufficient experimental data to deduce probability values depending on the robot state and environmental features.

Expert knowledge The missing knowledge concerning robot and surface interaction does not only affect the model description. Even for experts it is impossible to predict whether the robot is able to handle a given patch of surface easily\(^1\) or whether it will fall down. Therefore, well-known learning methods like reinforcement learning for the estimation of risk values cannot be applied to this problem.

Safety measures Existing safety measures for standard ground vehicles will be useful, but not satisfactory for this application. Well-known strategies like obstacle avoidance or speed limitation can be applied regarding macro structures like overlapping structures or gaps. Other measures to reduce the different risks – a climbing robot like CROMSCI is exposed to – cannot be derived from existing approaches and have to be examined, developed and tested.

Safe state A common safety measure for all kind of robots is an emergency stop, which transfers a standard ground vehicle or a robot manipulator arm to a safe state. As mentioned before, this is not suitable for a climbing robot using negative pressure adhesion. Here a safe state cannot be achieved by the robot itself because of the strong influences of the environment on robot safety. If the robot tries to drive over a rough surface area and detects problems (e.g. via a reduced downforce), it is not safe to stay at this position. In this case the robot should drive back to the last known 'safe' location or has to initiate other measures. Therefore, a safe state exists only in combination with certain environmental characteristics.

Environmental perception Last but not least, the robot is confronted with a limitation concerning the perception sensors. Because of its nature the robot has to be as light-weighted as possible. Each additional gram has to be lifted up the wall and increases abrasion of wheels and sealings, robot tilt, needed adhesion and driving

\(^1\)Of course, this can be said for very even (safe) and also for very rough (unsafe) structures. But everything in between, which is the common application of the robot, is hard or impossible to determine.
forces. Therefore, such systems are highly integrated combining all necessary sensors, actuators, controllers and other hardware components. It is not possible to carry multi-sensor arrays or high resolution distance sensors with a weight of several kilograms (compare section 2.1.5). The sensor system has to be as small and light-weighted as possible, which makes it necessary to get as much information as possible from the internal sensors.

Because of these limitations the creation of a safe-navigating climbing robot is a great challenge, which cannot be managed by classic safety concepts. Thus, the gap of knowledge and limitations has to be closed otherwise as postulated in thesis 2 to 4:

### Thesis 2
Risk estimation for a wall-climbing robot requires a control structure, which supports internal assessments and an analysis of the system's state. This estimation has to be used online as basis for compliant safety measures.

### Thesis 3
Safety measures have to be used on all stages of control from basic closed-loop controllers (in general, only risk reduction strategies) up to high-level control strategies to reach a tolerable risk level. These measures have to be dependent on the internal state of the robot, the current environmental situation and the desired task.

### Thesis 4
Offline analysis techniques have to be used to identify failures and possible causes in the development phase. They have to be the basis for structural (hardware) measures, but also for the development of additional software, which adapts the robot system on duty.

Within the scope of this thesis, different aspects, which can also be found in the documents presented by Kelly and Stentz (see section 3.2.3), will be taken into account: Environmental detection, online safety analysis and correct reaction of the robotic system. In a first step the hazards have to be examined and a fault tree analysis is used to identify causes and ways of handling (chapter 5). During all stages of development the limitations of the robotic system have to be kept in mind: Low payload and mounting space, no collection of statistical data and missing interaction model or expert knowledge of the dynamic system. Therefore, the present approach makes use of the internal state and the controller
reactions of the robot depending on its environment and extracts information from it. Because of missing expert knowledge this has to be done via optimization and learning techniques. In the present case, a genetic algorithm is applied to setup a function to predict hazardous situations (chapter 8) and to perform actions, which prevent the robot from a drop-off (chapter 9). Foundation methods for risk reduction and avoidance need to be developed for robot motion (chapter 6) as well as for robot adhesion (chapter 7). It will be shown, that robot safety cannot be ensured or realized with one single measure and that several safety elements are needed.

In the next chapter, the safety concept itself will be presented. This includes a determination of different types of hazards based on a fault tree analysis and in which manner these hazards this could be handled.
5. Safety Analysis and Concept

As given in chapter 3, there exist countless hazards, analysis techniques and safety measures. The challenge is now to find a safety concept, which is able to deal with imprecise models and missing world knowledge as well as with a lack of failure data and limited perception capabilities. The main safety concept is divided into offline and online methods. The offline analysis methods will help to identify problems, hazardous states and their causes. Corresponding safety measures can either be realized in hardware or implemented in software – depending on the given possibilities and constraints. In a common robot vehicle most of the hazards can be foreseen, analyzed and handled offline, as described in section 3.2.3. But, this analysis is not sufficient for climbing robots and additional online methods will be necessary in this special case. Therefore, the safety concept consists of different steps beyond the offline analysis and basic safety measures:

Step 1: Offline risk and safety analysis
The start point is an offline analysis to determine hazards and possible causes. This step also allows a classification of risks concerning their predictability, which makes it possible to distinguish between risks what can be handled offline – with measures which can be realized straight-forward – and hardly predictable risks that have to be taken into account online. This step uses fault tree analysis to identify errors and causes.

Step 2: Basic safety measures in hardware
Based on the results of the offline analysis, safety measures related to hardware are realized to ensure a minimum level of safety and operability. These measures include e.g. additional hardware components like a security rope or environmental sensors for obstacle detection, but also improvements of the existing hardware (e.g. weight reduction or enhanced weight distribution, more robust sealing and wheel rubber materials).

Step 3: Adapted software and controller measures
Additionally, further software elements like closed-loop controllers or perceptual software components need to be developed or existing elements need to be enhanced. This mainly includes elements for risk reduction like an improved chamber pressure controller or methods, which improve navigation capabilities of the robot. Nevertheless, also risk avoidance methods like evasion maneuvers in case of obstacles need to be considered and realized.
Step 4: Online estimation and prediction of risks As the detailed offline analysis will show, not all hazards can be foreseen and described sufficiently. Therefore, an online estimation and analysis will be necessary to allow a reaction of the system to current dynamic events. Beside the estimation itself this step therefore also contains additional software measures like driving strategies triggered by the results of this analysis.

In the next section, risks will be examined using a standard fault tree analysis. Based on the acquired results, different requirements will be derived related to individual components of the robot, which should be fulfilled to improve safety. This includes hardware measures as well as online safety measures in software activated during robot’s operation (section 5.2).

5.1 Offline Safety Analysis

At first, an offline analysis has to be performed to get an idea of possible hazards and corresponding applicable safety measures. As shown in chapter 3, there exist many events, which could lead to a damaged or even broken robot. These events have to be decomposed to a couple of primary events to know the sources and the amount of their influence on the system. The following description will concentrate on the main threat of a robot drop-off (which is the hazard with the highest severity level) and examine potential causes and involved components.

5.1.1 Fault Tree Analysis

Because of its universal nature, fault tree analysis is used in the present case. Further analysis techniques are summed up in appendix A.5.2. This analysis method can be used to identify and prevent undesirable events. It uses graphic symbols to describe and visualize the procedure and starts at the unwanted event [Dhillon2003]. Events, which could cause the hazardous event, are identified and connected by logical operators like AND, OR, XOR, and NOT. Error events are subdivided into different sources until an event has been reached, which either can not be divided or should not be examined more detailed. Therefore, the resulting fault tree is a logical structure, which connects the main event to the precipitating primary events. It allows the calculation of a probability of occurrence of the main event, if these values exist for the leaves. Each malfunction is split into its causes, until the basic causes are determined or an event has been reached, which can not be decomposed (e.g. human failure). The four event nodes and subtree icons for fault tree analysis are:

- **Malfunction**: This event is also know as composed event and results from a combination of underlying events. Malfunctions have to be split into their components for further analysis.

- **Primary event**: Primary or basic fault events do not have to be subdivided because they are either well-known or their failure probabilities, frequencies and modes are known.
5.1. Offline Safety Analysis

Secondary event: For this event, the underlying causes are left unexamined. This can happen because of missing failure data, so that a more exact description is not possible, or because a further subdivision is not necessary.

Trigger event: Trigger events are no error sources, but can lead to errors in combination with other events.

Subtree: A subtree is used as a place holder for deeper branches of the fault tree.

In this context it is also important to distinguish three different types of errors with a defect as a superset of failures and faults:

Defect  Abnormal behavior or irregularity of a component from its predefined characteristic. The task execution is disturbed, which leads to an error state of the system.

Failure  Malfunction of a component, which cancels the fulfillment of the desired task. A failure leads implicitly to a defect.

Fault  Inability of a component to perform the desired task. In this case the component has reached its limits e.g. because of insufficient dimensioning, incomplete design or external causes. A fault leads implicitly to a defect.

![Fault tree for a sensor malfunction](figure5.1)

Figure 5.1: Fault tree for a sensor malfunction (figure taken and adapted from [Visinsky1993]).

In literature, descriptions and example fault trees for different general events like a sensor, motor or power failure, can be found. Figures 5.1 and 5.2 show examples for general causes of sensor and motor failures as they are given in the work of Visinsky [Visinsky1993]. In general, it has to be considered that failures affect each other. A power loss of a component can be caused separately e.g. by a broken cable as well as from a global power...
failure affecting all or a large number of electronic elements. This global interaction is not described explicitly here because this should be obvious and does not affect a single component solely. As shown in figure 5.1, a motor failure can also cause a sensor system to fail if this is a system with active motion (e.g. a movable laser scanner or a pan-tilt-camera) or uses an internal motor (e.g. in 2D laser scanners for rotating the mirror). Vice versa some types of motors also include motor encoders to detect the rotational speed or the current rotational position of the motor axis. Thus, a malfunction of this sensor can cause the motor controller – and as a result the motor itself – to fail. In figure 5.2 the subtree encoder failure has been added, which is similar to figure 5.1, but leaves out incorrect calibration and – of course – a motor failure.

Figure 5.2: Fault tree for a general motor malfunction including a failure of the encoder sensor (figure taken and adapted from [Visinsky1993]).

As mentioned before, the most severe hazard is a robot drop-off, which should be avoided under all circumstances. Chapter 3.4 gave an overview on potential hazards leading to this incident and how they can be described, but so far there is no point of action given to counter these hazards. In figure 5.3 the causes of such a drop-off are shown. Corresponding to table 3.3, four errors can cause this incident, which will be described in the following sections: Robot slip and tilt (which have been combined here), a leaky sealing\(^1\), a power failure or an electronic malfunction.

\(^1\)This includes also the driving into a hole which could also lead to a damaged and therefore to an untight sealing rubber.
5.1. Offline Safety Analysis

Regarding a complex electronic system like a climbing robot it is obvious that malfunctions of the electronical components or of the power supply lead to problems. These errors are omnipresent and are independent from the current execution task and actions of the robot. As shown in figure 5.3, a power failure of the system can be caused by a defect of the power generator, of the connection cable or its adapter plug, or by an overstraining caused by a high power consumption of the internal components. In general, these causes can be described by basic fault events for which error rates like mean time to failure \(2^{nd}\) (MTTF) can be determined. Reasons for these errors can be design failures, manufacturing problems (also known as infant mortality), random failures and wear out. Table 5.1 gives a very rough view on common reliability values of hardware components. It is not possible to sum up the wide range of products, different manufacturers and non-standard specifications, but the table should allow a general view on the magnitude of these failure rates. The MTTF ranges from only 1 000 hours for some DC motors – mainly because of wear of the sliding contacts (brushes) – up to thousands of years for simple semiconductor devices.

Other obvious causes for a drop-off are malfunctions of the robot electronics, which include motors and valves, sensors, the communication system and the control computer. Figure 5.4 sums up the underlying errors. Motor and sensor failures can be subdivided according to figures 5.2 and 5.1.

---

\(2^{nd}\)The MTTF is used for components, which are not repaired or reused and which have to be replaced. Otherwise one speaks of the mean time between failures (MTBF), although both terms are often used interchangeably.
Table 5.1: Common failure rates given as *mean time to failure* or *mean time between failures* of electronic and mechanical components.

<table>
<thead>
<tr>
<th>Components</th>
<th>MTTF / MTBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industrial PCs(^a),(^b)</td>
<td>100 000 – 200 000 hours</td>
</tr>
<tr>
<td>Hard disk drives(^c)</td>
<td>1 000 000 – 1 500 000 hours</td>
</tr>
<tr>
<td>Laser scanners(^d)</td>
<td>50 000 – 70 000 hours</td>
</tr>
<tr>
<td>Motors and gearboxes(^e),(^f)</td>
<td>1 000 – 20 000 hours</td>
</tr>
<tr>
<td>Vacuum engines(^g)</td>
<td>1 500 – 10 000 hours</td>
</tr>
<tr>
<td>Semiconductor devices(^h)</td>
<td>&gt; 20 000 000 hours</td>
</tr>
<tr>
<td>Pressure sensors(^i),(^j)</td>
<td>300 000 – 500 000 hours</td>
</tr>
<tr>
<td>Power supply(^k)</td>
<td>100 000 – 400 000 hours</td>
</tr>
</tbody>
</table>


It has to be mentioned that a motor defect does not necessarily lead to a drop-off and that it is not always possible to detect these malfunctions. In the case of CROMSCI, the chamber valves e.g. are build up by using stepper motors, which do not have any encoders. Thus the motor controller does not get any response from the motor itself whether the desired position has been reached or not. By keeping this in mind it is valid to assume the worst case and define that a blackout of such an important component could lead to a robot drop-off. The same can be said for the pressure sensors or load cells whose values are used by the closed-loop controllers and could produce a maloperation of them, if they fail or are to noisy. Hazards which do not necessarily cause a drop-off like collisions and a *driving into a hole* are not handled here explicitly. Also *mechanical defects* at the robot chassis are not mentioned explicitly, since their chance of occurrence is neglectable compared to the malfunctions caused by dynamic effects between robot and environment.

### 5.1.3 High Dynamic and Hard Predictable Risks

The second type of malfunctions are dynamic errors caused during operation, which cannot be described easily. These problems occur if the robot is in motion and interacts with its environment. Therefore, they are much more difficult to formulize. Figure 5.5 (left) shows the fault tree for the *malfunction of the sealing*\(^4\). Beside not further specified causes (e.g. a wrong target pressure value by the user) the sealing can fail either because of a defect in the pressure closed-loop controller program or if the sealing is damaged and the pressure controller is not able to achieve the desired target value. A deeper look concerning these causes and how they can be prohibited has not been taken so far.

\(^3\)Depending on the component the manufacturers either give the mean time to failure (MTTF) or the mean time between failures (MTBF) as reliability measure.

\(^4\)Defects of the supporting electronical components like pressure sensor, valve or pressure pump are summed up at the fault tree given in figure 5.4.
Figure 5.4: Underlying events, which can cause an electronic malfunction. Foreseeing environmental sensors are not considered here, because they do not affect a robot drop-off directly.

Figure 5.5: Fault trees with default controllers for sealing and chamber pressures.

Although the pressure controller may not be hundred percent failsafe it is assumed that – if it is designed well and the parameters are determined carefully – the residual risk of this defect can either not be lowered without high costs or it is neglectable compared to the other causes of a leaky sealing. The same counts for the not further specified
causes whose probability is not really ratable. Therefore, one has to focus on the causes leading to a damaged sealing to reduce the overall chance of this fault tree to come true. Such a damage of the sealing is only problematic concerning a drop-off if the pressure controller (in combination with valves, sensors, pressure reservoir and pressure pump) is not able to balance this defect and to generate the desired sealing pressure for leak tightness. The idea now is to find underlying causes and corresponding safety measures to lower the probability of their occurrences. In figure 5.6 the updated fault tree of a leaky sealing is given. This description handles only internal causes including different kinds of damages of the sealing rubber or its coating. A basic leakage or leakages because of an uneven surface are not considered here, since the adhesion and pressure systems should be dimensioned sufficiently to handle these permanent and unavoidable disturbances.

Figure 5.6: Fault tree to find the causes for leak sealings and suitable counteractive measures.
5.1. Offline Safety Analysis

Figure 5.6 explores the causes of a damaged sealing and describes the influence of additional safety measures to prevent damages. The sealing can be damaged if it is ripped by protrude obstacles like a sharp concrete ridge or edge, if the service by a user failed (e.g. if the sealing is not mounted in a correct manner) or if it is worn-out. The last case can be caused by the permanent abrasion of the sealing’s coating in combination with missing or failed maintenance. Here only operation guidelines, which prescribe regular inspections of the sealing and a more robust sealing material might reduce the chance of this hazard.

A more interesting subtree represents the causes of a ripped or penetrated sealing, since this is no permanent event, but still depends on the environment and the interaction of the robot. It should be obvious that such a damage at the sealing cannot occur if the concrete structure being able to cause it, can be detected and avoided. Therefore, the subtree is extended with a detection part responsible for the foresighted analysis of the ground and with a reactive part, which contains suitable safety measures to evade or handle the detected hazard. Although – of course – also these measures might fail they are necessary to reduce the chance for a ripped sealing and therefore for a drop-off of the robot. It has to be mentioned that a malfunction of a foresighted sensor like a laser ranger for the detection of hazardous environmental features does not necessarily lead to an instant drop-off just as well as an imprecise localization of the robot and the feature. But, if these failures are not handled by the system and other conditions like the robot approaches a sharp edge are fulfilled they could lead to severe damages because of missing or erroneous reactions.

The second dynamic cause of a robot drop-off, as shown in the fault tree in figure 5.3, is robot slip/tilt. So far, only basic closed-loop controllers as developed by Hillenbrand are considered to balance out the negative pressure inside of the chambers and the overall downforce [Hillenbrand2009]. According to figure 5.5 (right) slip or tilt occur, if the cascaded closed-loop controllers reached their limits because of high leakages or in case of internal controller defects. Again, failures of hardware components like circuit boards, valves, or pressure sensors are already considered in figure 5.4 and therefore not taken into account here.

Since slip and tilt are the hazards with the highest probability (and therefore the highest risk) their causes have to be examined in detail to find methods to fix them or measures to reduce the chance of their occurance. Based on the descriptions before, it can be assumed, that the probability of a defect of the chamber or force controllers in figure 5.5 (right) is relatively low. Also the vacuum generator might not be replaced by a stronger one because of payload and mounting space restrictions. So the most effective way to reduce the chance of slip and tilt would be to append further elements like more foresighted measures and strategies to avoid situations of high leakages causing a controller fault. Figure 5.7 sums up the additional causes and shows necessary work on the analysis, detection and handling of these risks. In this view wheel slip is added as additional source of robot slip whose chance of occurance has to be reduced via an improvement of the hardware or extra control components. To reduce the probability of high leakages during interaction between robot and environment, control elements have been added which try to keep the robot away from surface areas, which cause these dangerous leakages. In fact the probability for a controller fault is reduced by lowering the chance that the robot

5The subtree of the malfunction reaction failed can be found in figure 5.7.
5. Safety Analysis and Concept

Figure 5.7: Causes for slip and tilt of the robot: If the robot drives on a patch of surface, which causes too high leakages, the robot will be lost. The drop-off is a concatenation of different malfunctions.

reaches a critical situation. The closed-loop controllers can now be seen as one part of the general safety measures. Again, malfunctions of the included emergency strategies can be caused by an deficient detection of the features or of missing or faulty reactions on these events. In general, the adding of further measures and elements of course causes new possible error sources. Therefore, these components have to evaluated whether they are really useful and lower the risks or not.

Based on the fault tree analysis, it is now possible to derive corresponding safety measures and extract certain requirements the system must fulfill.
5.2 Safety Requirements

As shown before, full model knowledge for a complete and sufficient description of the interaction between robot and environment cannot be obtained. Therefore, the gap between missing and needed knowledge has to be closed by the safety concept as presented in this section. This will be done via requirements based on the results of the FTA. Because of the much higher probability of the occurrence of external hazards (in contrast to internal errors) malfunctions of the electronics, sensors, actuators, mechanics, and power supply are not considered here explicitly. Nevertheless, the robot hardware has to be in a state that the system is operable.

5.2.1 Design and Hardware Requirements

Hardware measures have to be considered in the design and construction phase of the robot or after a safety analysis. The goal of these measures is to ensure a minimum level of safety to enable the system to perform the desired tasks. Beside construction and realization of hardware components these measures also include controllers and electronics, which are directly linked to them. All these measures have in common, that they require changes in the robot construction or that they are mandatory for any operation. If the weight does not correspond to the number of adhesion chambers or to the suction area the whole system will fail. But, with a careful look on the desired parameters and their influences it is possible to set up a functioning system. Only dynamic elements, which occur during operation cannot be handled easily or by adaptations of the hardware.

Requirement 1 – Hardware components must fit

This requirement results from all fault trees, which contain robot hardware (e.g. from the fault tree given in figure 5.4 describing the sources of an electronic malfunction). It proclaims that all single components of the system have been optimized regarding their interaction and their influence on the system behavior. In general, the weight is the main opponent of a climbing system. The lower the total weight the less adhesion forces are needed, which allows higher leakages, less suction area, and reduces wear of wheels and sealings. Therefore, the weight influences many nodes of the shown fault trees: If the adhesion forces are lower e.g. the drive motors need less power for motion, which reduces the probability of high power consumption (fault tree in figure 5.3) or the effect of high leakages (figure 5.7). Therefore, lightweighted materials like glass fiber or carbon fiber have been used and suction engines, drive motors, and other parts have to be dimensioned carefully to find the optimum between needed power and weight. Other important requirements concerning the hardware are a good mass distribution (as close to the wall as possible to reduce robot tilt) or a low wear of high mortality parts by using optimized materials. Also a security rope as final safety measure is mandatory as it is required by law.

Requirement 2 – The supply has to keep sufficient reserves for critical situations

All supplying components must be dimensioned in a way, that they keep sufficient reserves in situations of high consumption. This includes energy supply of the robot, as depicted in the fault tree in figure 5.3, but also the onboard vacuum engines for the negative
pressure system and the pressure pump for the sealing (fault trees in figure 5.5). If one of these elements is at its limits (trigger event) it does not necessarily lead to a drop-off, but it reduces the chance that a critical situation can be handled. High consumptions can be caused by large leakages at the sealings (affecting the suction engines) or by small damages of the sealings so that they lose air. But, also the power consumption can reach its limits e.g. if the wheels are turned and driven with high downforces so that they need more power for motion. Finally, the computational power of the embedded onboard computer has to be sufficient to process the sensor data and to generate suitable motion commands and control values in time. Of course, there is an upper limit concerning size and weight, which makes it much more difficult to find optimal components as described by requirement 1.

Requirement 3 – Robust controllers are necessary

The third foundation of minimum safety are well-designed closed-loop controllers for adhesion, motion and sealing pressure. This mainly addresses the probabilities of controller defects as depicted in the fault trees from figures 5.6 and 5.7. These controllers must be able to balance out disturbances fast, robust, and effective – and in the ideal case also oscillation-free control as described by [Hillenbrand2009]. Of course, also the sensors and actuators have to be precise and fast enough to allow an optimal closed-loop control. It has to be kept in mind that the additional safety measures have no chance to engage if the adhesion or pressure controllers do not operate correctly. If e.g. the force controller produces high oscillations it might be possible that the robot drops off because of a too low adhesion force during the phase of underswinging. The opposite case is not that safety critical, since the adhesion remains assured if the downforce is higher than desired. In this case the possible robot motion might be reduced if the force is too high.

Requirement 4 – Problems in control components have to be identified easily online

Although the fault trees allow a good view on the safety-related components of the system it is essential to identify errors and their causes also during runtime. Therefore, suitable tools and a software framework are needed supporting the detection of erroneous system states, defects in single components, and a trace back of the underlying reasons. Related to the fault trees this requirement demands that it should be possible to analyze each leave of the trees during runtime if it is operating normally or if any malfunctions or disturbances occur. It should be also possible to adapt important system parameters online to be able to test different setups, but also to influence individual components if it becomes necessary. In fact this requirement will not avoid a drop-off of the system, but it helps to find the causes and therefore to avoid this situation next time.

Requirement 5 – Limit operation time for periodical check ups

A general problem for that kind of climbing robot is the abrasion of the sealing’s coating and of the wheel’s rubber. In the worst case, the robot can fall down because of a damaged sealing or get stucked at the wall if the wheels are worn off. The operation time of the system has to be limited depending on the expected wear of these high mortality parts. A
check up of these components should be requested before or after the task execution. By regular check ups the problem of wear can be reduced to a minimum as it can be derived from the fault tree in figure 5.6. Nevertheless, the optimal time for maintenance is hardly to determine because of changing friction, roughness or other factors.

5.2.2 Online Safety Requirements

Beside the hardware requirements also a couple of online requirements for robot safety can be read out of the fault tree analysis. Considering the hazards given in table 3.3 and the leaves of the fault tree in figure 5.3 the basic system has to be enhanced using additional controllers, detectors and classifiers. Some of them are straight forward because their causes and influences on the system are describable more or less easily, whereas some others are hard to describe and not easy to realize. The handling of these requirements will be part of the following chapters.

Requirement 6 – No wheel steering on the spot

This requirement helps to reduce wheel slip, which effects the chance of robot slip/tilt, as depicted in the fault tree in figure 5.7. A turning of the wheels on the spot would change the friction characteristics of the wheel rubber from static to dynamic friction. The result is a lower friction value, which could let the robot slip down. There is also higher wear and even a higher chance of damaging the wheel’s rubber. Of course, this requirement can also be applied to standard ground vehicles although it does not have such importance as for climbing robots.

Requirement 7 – Wheel slip in motion has to be reduced

As presented before, wheel slip is a serious problem concerning robot slip, but also with respect to wheel abrasion and positioning errors. Although it cannot be avoided completely during motion it has to be minimized to reduce these negative effects and to reduce the chance of robot slip (fault tree in figure 5.7). As mentioned in section 3.4.5, wheel slip occurs if a force has to be transferred via the wheel. The amount of slip depends on the difference between transferable and acting force. Thus, the acceleration has to be limited via a kind of traction control system. This measure will not influence the system in a negative way, but reduce wheel slip if possible.

Requirement 8 – Shear forces between wheels have to be reduced

Another way of decrease wheel slip and therefore also decrease robot slip according to fault tree in figure 5.7 is the reduction of shear forces. These oppositional sideward forces between the wheels reduce the maximum transferable forces in rolling direction as described in section 3.4.5 and raise the chance of wheel slip. Also the wheel’s rubber would be worn out faster. The only way to deal with these forces is the usage of special controllers, which balance them out by slight changes in a wheel’s steering orientation and driving velocity.

---

6If the robot is able to work autonomously for a longer period of time such a check up is also possible during runtime after a certain covered distance by driving to a home position, at which the robot can be accessed and maintained by a technician.
Requirement 9 – Steering of wheels should be minimized

The fourth way to diminish wheel slip (fault tree in figure 5.7) and wheel abrasion is the minimization of wheel steering. Because of the two degrees of freedom of a steerable standard wheel there are multiple choices for setting up a certain steering direction. For a faster execution of steering commands and to avoid unnecessary wear of wheels and bearings, the underlying wheel control should choose the best motion commands.

Requirement 10 – Obstacles must be avoided

As presented in section 3.4.2, there are a couple of problems caused by collisions like damages at the robot chassis or robot stuck. Although a collision of the robot chassis with protrude obstacles may not be highly critical, especially a penetration of the sealing by a sharp edge can become dangerous and lead to a robot drop-off. Therefore, the detection and evasion of positive and negative obstacles are important to safeguard the robot from damages as it is shown in fault tree figure 5.6. But, negative obstacles may also cause high leakages, so their avoidance also reduces the chance of the robot to slip or tilt (fault tree in figure 5.7). To reduce the probability of these events protective elements in form of evasive actions are needed, which let the robot turn away from the hazard.

Requirement 11 – The robot has to slow down if obstacles are ahead

In a similar way it is also possible to reduce the impact of a collision by slowing down the robot. In fact, this measure supports the evasion process, since it gives the user or the automatic safety measures more time to react on the threats. This requirement includes positive obstacles in the same way as negative obstacles so it also reduces the chance of the robot to drive into a hole (and therefore of robot slip/tilt as described in fault tree figure 5.7). The idea is to slow down the robot up to a full stop if a minimum distance has been reached. Of course, it also reduced the chance of damages at the sealings if the robot stops in front of sharp edges, which could destroy the sealing rubber and cause a drop-off (fault tree in figure 5.6).

Requirement 12 – Downforces have to be balanced out equally

As described before, robot tilt is one of the primary causes for a drop-off. It becomes obvious that – to avoid robot tilt – the downforces have to be balanced out equally among all three wheels (compare section 3.4.7). Such a balancing does not effect the probability of high leakages as given in the fault tree in figure 5.7, but it is able to reduce the effect and therefore the severity of these leakages. Additionally, an optimal distribution of the downforces also has a positive effect on the wheels itself, since the lower wheels would be pressed stronger against the wall than the ones on the top without this measure. This downforce difference would lead to a higher wear of the bottom wheels and to a stronger squashing of the wheel rubber. Also the driving forces would be used suboptimal what could promote wheel slip.

Requirement 13 – Leak chambers must be cut off

Chambers, whose leakage is larger than their maximum valve area, cannot be supported by the adhesion system, since they lose more negative pressure than producible. Such a

\[\text{Of course, there exists a limit, which makes it impossible for the downforce controller to balance the forces. But in general, the chance of robot tilt is reduced by this measure.}\]
leak chamber would let the whole system to fail because its leakage affects the reservoir chamber, reduce its negative pressure and thus lower the overall downforce. Finally the robot would fall down. Leak chambers have to be identified based on their amount of leakage and shut down by closing their valve, which reduces high leakages at the adhesion system as it becomes necessary in figure 5.7. Of course, a deactivated chamber does not provide anything to the robot’s adhesion and involves the risk of promoting robot tilt or slip. But, the same counts for a leak chamber, which is not cut off from the adhesion system. Therefore, this measure will help to ensure a certain portion of downforce, but has to be used carefully. It is no magic bullet – especially the period and order of a reintegration of deactivated chambers has to be chosen well. But, also the maximal number of deactive chambers and their positions have to be considered (so the reaction must not fail as demanded by the fault tree in figure 5.7).

**Requirement 14 – Reintegration process of chambers must not stress the adhesion system**

This requirement is important regarding the reintegration process if one or more chambers have been cut off from the adhesion system (compare description and point of action of requirement 13) and will help to avoid robot slip and tilt. Of course, the chambers have to be tested one after the other to limit the affect on the adhesion system. But, a simple periodical testing is not sufficient to have the chambers reintegrated as soon as possible with a minimum of stress, which makes additional measures necessary.

The presented requirements and measures have an effect on the robot’s safety and should be realized. Some basic elements and requirements like pressure and force controller or the avoidance of wheel steering on the spot have already been introduced or mentioned by [Hillenbrand2009]. Most of them have been updated within the scope of this thesis whereas other measures have been developed completely new. These requirements have in common that they consider more or less well-known effects on the robot. Of course, they are important for robot safety, but not sufficient.

### 5.2.3 Additional Requirements for Hard Problems

The requirements, which have been presented so far, reduce the overall risk via additional controllers and measures. But, they are not able to consider the whole state of the robot, its adhesion system and the surrounding. Therefore, one main point in the question of robot safety is based on the identification of hazardous situations – e.g. of an upcoming drop-off – to be able to react on them. Thus, upcoming hazards have to be detected beforehand and handled in a correct manner, as given in the following two requirements related to the detection and reaction subtrees in figure 5.7.

**Requirement 15 – Hazardous situations must be detected**

If the robot comes in a hazardous situation, which could not be expected or foreseen, the system has to be able to identify this situation as crucial early enough to initiate counteractive measures. General indicators for these situations might be the adhesion force, the divergence of the downforce point to the robot center, the internal controller states or estimated leakage values. Of course, the detection has to be robust to notice all critical situations, but also to avoid false positives. If an upcoming crucial situation has been detected, a fitting reaction needs to be executed.
Requirement 16 – The robot must react properly to current hazards

The early detection of a critical situation is only one side of the medal. Moreover the robot has to react in a correct manner to the identified threat otherwise the detection would be useless. Triggered by a risk prediction the robot has to perform actions like a full-stop, backward driving, or the driving to a safe position to avoid a drop-off. As a result, the robot remains adhered to the wall.

5.3 Safety Concept

The identification of hazards and their causes as well as the determination of requirements are first steps for enhancements of the navigation safety of a robot. The next step is now to think about the realization of these requirements and to develop corresponding safety measures and online analysis methods. The general safety concept applies a combination of closed-loop controllers and behavior-based control elements. Figure 5.8 gives an overview on the different components and levels.

![Figure 5.8: General view on the safety concept including hardware measures (blue), basic safety measures (green), avoidance of obvious risks (yellow) and methods of risk estimation and prediction (red).]
The fundament for a safe system is formed by hardware components (blue), which meet the given requirements and provide a certain amount of operability and operation safety. Related to the fault trees these components mainly address the subtrees of a power failure (figure 5.3) and an electronic malfunction (figure 5.4). The next higher layer consists of the closed-loop controllers, which will control the robot hardware components e.g. in terms of low-level controller on embedded DSP circuit boards or implemented as software modules on the control computer. The idea is to make use – even on this basic closed-loop control level – of the concept of a behavior control network in combination with closed-loop controllers to join the characteristics of both worlds. Beside an easier bottom-up development, a better online analysis and further advances, the system will use behavioral meta data for the detection and prediction of critical situations as depicted by the risk estimation module. Also the complete upper robot control components will be realized as behaviors, which control the robot motions and evade or reduce the risks by executing special motions or other actions. In between software elements are located performing online measures – e.g. in terms of motion adaptions – as well as detection of obstacles and risk estimation based on the state of the adhesion system. These elements correspond to the new control and detection components shown in the fault trees of figures 5.6 and 5.7. It has to be mentioned that the existing and retrievable system and interaction knowledge of the elements decreases in figure 5.8 from left to right. So the adhesion control elements and corresponding analysis and reactive components (red) are all other than straight forward in development.

The next chapter will introduce the advanced motion control components, which have been develop to meet some of the presented requirements related to the drive system (green modules in figure 5.8). Chapters 7 and 8 will present the components for adhesion control and a method used to predict risks like a robot drop-off. Corresponding safety measures and the detection of obvious hazards like obstacles follow in chapter 9.
6. Advanced Motion Control for Safe Navigation

As given by the requirements in chapter 5.2.2, special components are needed to answer a couple of safety-critical demands. Some of these requirements address the basic vehicle locomotion and will be considered during the kinematic calculations or as additional controllers. This chapter will deal with these aspects on the closed-loop control layer, and will introduce the advanced motion control (AMC) components, which contribute to robot’s safety regarding navigation [Marx2009, Schmidt2012]. The task is to fulfill the given safety requirements, since the robot should not only adhere to the wall, but also move on it. This includes safety measures like a traction control system or a controller to reduce sideward shear forces. The benefits of these methods concerning robot safety will be shown in experimental results at the end of each section. Figure 6.1 gives an overview on the used and developed components. The basic motion control system can be split up into three different layers:

**Planning** The planning layer on top contains the forward and backward kinematics of the omnidirectional drive and a module for adaptions of the control values. It transforms desired robot motions into necessary commands for the three wheels, as already described in section 2.1.1, and executes special adaptions, which will be shown in upcoming section 6.1.

**Reactive** The reactive layer in the middle adapts desired orientations and velocities to reduce shear forces. Section 6.3 deals with these additional closed-loop shear force controllers. The resulting values are given from the PC via a CAN-bus connection to three DSP boards.

**Basic** The basic layer is based on the three DSP boards. Each circuit board contains basic closed-loop motion controllers, which are responsible for one steerable driven standard-wheel. Again, each wheel unit consists of a load cell and two motors for wheel steering and driving, as depicted by the symbols from left to right. The onboard controllers adjust a desired position or velocity, and perform the traction control, as it will be described in section 6.2.
Table 6.1 sums up the differences of the three layers depending on their execution hardware, control behavior and whether they use local data or have a more global view. This distinction into PC and DSP controllers is necessary because of a missing inter-DSP-communication, which makes it impossible to develop controllers on a DSP board relying on sensor values from other DSPs. In case of a single wheel, the corresponding DSP is responsible for this wheel only and does not know anything about the other two wheels.

---

**Figure 6.1**: Structure of the AMC based on two software layers, DSP boards and several hardware components (for symbol descriptions see table A.3 in appendix A.2).

**Table 6.1**: Three layers of motion control and their general differences (compare figure 6.1).

<table>
<thead>
<tr>
<th>Layer</th>
<th>PC/DSP</th>
<th>Realtime</th>
<th>Closed-loop</th>
<th>Local Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>planning layer</td>
<td>PC</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>reactive layer</td>
<td>PC</td>
<td>yes&lt;sup&gt;a&lt;/sup&gt;</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>basic layer</td>
<td>DSP</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

<sup>a</sup> The reactive motion control layer runs with a short cycle time of about 10 ms, but does not operate in real time in a strict sense.
6.1 Ambiguity and Boundary Conditions

Based on the kinematic calculations from section 2.1.1, desired steering and turning commands for the wheels can be determined. But, as in most cases, the results from the inverse kinematic are not unique. It is obvious, that there exists an unlimited number of possible wheel orientations due to the fact that the wheel domes can turn endlessly. Furthermore, it is possible to rotate in both directions to reach a specific position. To handle these ambiguities, boundary conditions have to be applied. Common requirements are e.g. minimized joint motion or energy optimization.

6.1.1 Avoid Wheel Turning on the Spot

In this case, requirement 6 of section 5.2.2 provides some boundary conditions since it interdicts wheel steering on the spot. If the desired wheel velocity \( v_{\text{des}}^{W_i} \) is zero, the wheel does not contribute anything to robot motion. In this case the target steering angle \( \varphi_{\text{des}}^{W_i} \) is set to the current one \( \varphi_{\text{act}}^{W_i} \) (equation (6.1)), which keeps the wheel dome in its current orientation. Otherwise it uses the desired angle \( \varphi_{\text{des}}^{W_i} \) calculated by the kinematic:

\[
\varphi_{\text{des}}^{W_i} = \begin{cases} 
\varphi_{\text{des}}^{W_i}, & \text{if } |v_{\text{des}}^{W_i}| > 0 \\
\varphi_{\text{act}}^{W_i}, & \text{else}
\end{cases} \tag{6.1}
\]

The benefit is the reduction of unwanted wheel slip even in cases, if the robot only steers, but does not move. This may happen, if the robot drive system transfers from straight driving to full turning. An example for the benefit of this measure is given later on in figure 6.4.

6.1.2 Minimize Steering

The second requirement, which is managed by the kinematic system, is the minimization of steering by adapting the values from the inverse kinematic (requirement 9). Robot safety will be enhanced due to a reduction of wheel wear and an increased lifetime of bearings and sliding contacts. Additionally, the robot will be able to react faster on steering commands, which will reduce the occurrence of unwanted shear forces (what will also reduce robot traction as given in section 6.3).

The first option to reduce steering is to turn the shortest way depending on the current wheel dome orientation. The resulting motion can be \( \pm 180^\circ \) in the worst case. The second option is to additionally inverse the wheel’s rotation direction if necessary, which lowers the worst-case to \( \pm 90^\circ \) turning. To find the shortest turning angle the current steering value \( \varphi_{\text{DSP}}^{W_i} \in \mathbb{R} \) from the DSP board has to be evaluated to get the current amount of overturning \( \Phi_{\text{act}}^{W_i} \in \mathbb{N} \), as given in equation (6.2), and the current angle \( \varphi_{\text{act}}^{W_i} \in [-\pi, \pi) \) (equation (6.3)). This overturning is possible due to the mechatronical design of each drive unit being able to rotate without limits. The embedded motor encoders work incremental and count the motor turning in small ticks, which are converted to the real degree of turning. The addition of \( \pi \) in equation (6.2) is necessary to reach the desired range of the angle \( \varphi_{\text{act}}^{W_i} \). Equation (6.4) shows, in which way the DSP value \( \varphi_{\text{DSP}}^{W_i} \) can be constructed based on overturning \( \Phi_{\text{act}}^{W_i} \) and current angle \( \varphi_{\text{act}}^{W_i} \).
Equations (6.5) and (6.6) show the necessary adaptations of the velocity and steering values for a wheel. Similar to the current angular value, also the desired steering angle $\varphi_{\text{des}}^{W_i}$ lies in the range of $[-\pi, \pi)$.

The distinction in equation (6.6) between a negative and a positive desired steering angle is to keep $\varphi_{\text{act}}^{W_i}$ within the same range. Both equations (6.5) and (6.6) use function $|\alpha, \beta|^\circ$, which describes the absolute shortest angular distance of the two angles as given in equation (6.7) with $\alpha, \beta \in [-\pi, \pi)$. Figure 6.2 illustrates the different wheel turning options.

\[
\begin{align*}
\varphi_{\text{act}}^{W_i} &= \left\lfloor \varphi_{\text{DSP}}^{W_i} + \pi \right\rfloor \div 2 \cdot \pi \\
\varphi_{\text{act}}^{W_i} &= \varphi_{\text{DSP}}^{W_i} - \varphi_{\text{act}}^{W_i} \cdot 2 \cdot \pi \\
\varphi_{\text{DSP}}^{W_i} &= \varphi_{\text{act}}^{W_i} + \varphi_{\text{act}}^{W_i} \cdot 2 \cdot \pi
\end{align*}
\]

(6.2)

(6.3)

(6.4)

Figure 6.2: All four different motion commands depending on the current steering position $\varphi_{\text{act}}^{W_i}$ (center wheel) to minimize wheel dome rotation.

Since the desired steering angle is within a limited range resulting from the kinematic calculation, the final value might be increased or decreased to turn the shortest way if the desired angle lies on the opposite half of the circle. Therefore, the final steering
6.1. Ambiguity and Boundary Conditions

Angle $\varphi_{W_i} \in \mathbb{R}$ is calculated according to equation (6.8). $\Phi_{act}^{W_i}$ is again the number of current wheel overturning. Figure 6.3 illustrates the correlation of the different values in an example: To turn the wheel from the current position $\varphi_{act}^{W_i}$ to the desired one $\varphi'_{W_i}$ on the shortest way (green), the desired value has to be increased by $2\pi$ which makes the final value larger than $\pi$. Otherwise the red turning arc would be executed.

$$
\varphi_{W_i} = \begin{cases}
\varphi'_{W_i} + \Phi_{act}^{W_i} \cdot 2\pi + 2\pi, & \text{if } |\varphi'_{W_i} - \varphi_{act}^{W_i}| > |\varphi'_{W_i} + 2\pi - \varphi_{act}^{W_i}| \\
\varphi'_{W_i} + \Phi_{act}^{W_i} \cdot 2\pi - 2\pi, & \text{if } |\varphi'_{W_i} - \varphi_{act}^{W_i}| > |\varphi'_{W_i} - 2\pi - \varphi_{act}^{W_i}| \\
\varphi'_{W_i} + \Phi_{act}^{W_i} \cdot 2\pi, & \text{else}
\end{cases}
$$

(6.8)

Figure 6.3: Example to handle ambiguous values in the best manner (shortest turning).

6.1.3 Experimental Results and Advantages

Both measures – no turning on the spot and minimized steering – affect robot safety in a way that they reduce wear of the wheels and dynamic friction. Additionally, the robot is able to reach its desired trajectory faster because the wheels have to turn only $90^\circ$ in the worst case compared to $180^\circ$ in the former setup.

Figure 6.4 depicts results of an experiment on the ground to proof the benefit of the avoidance of steering on the spot. Here, the real prototype CROMSCI has been pulled manually two times and the sideward forces have been measured. In the first case the wheel domes were rotating to simulate a steering on the spot (green graph). In average, the needed sideward force to pull the robot\(^1\) is 137.5 N. In contrast to that, the wheels were set fixed in a second test run (blue graph), which results in an average sideward force of 159.9 N. By considering these findings one can state that turning on the spot reduces the friction value to 86% of its maximum and raises the chance of robot slip. This effect is smaller if the robot is adhered to a wall because also the sealings absorb sideward forces, but, nevertheless, this is an easy, but effective way to reduce robot slip and to improve its safety.

\(^1\)It has to be mentioned, that only the bottom part of the robot with the drive units has been used here, which reduces the total weight of the system.
6.2 Traction Control System

As already mentioned, wheeled systems have a general problem of wheel slip (compare section 3.4.5). This problem increases if the robot is going upwards a vertical wall and has to overcome gravity. Regarding requirement 7, the reduction of wheel slip is one important method to reduce wheel abrasion and to improve robot movement. Beside general optimizations like the reduction of robot weight or an increased wheel grip, a traction control system (TCS) might be a good solution to reduce wheel slip. Modern traction control is a well known field in automobile development since its launch in 1987. The first traction control systems have been set on top of an already existing antilock brake system (ABS) to brake wheels selectively depending on the measured wheel slip. Other systems reduce the drive torque by influencing the motor controller itself e.g. via throttle flap, the amount of injected fuel or the ignition point [Reif2010]. Concerning mobile robots Zielinska et al. sum up different ways to estimate traction force and slip coefficient [Zielinska2010].

In contrast to a common automobile, CROMSCI does not have any passive wheels which could be taken into account to measure a rotary velocity as reference to detect wheel slip. Also the motor current cannot be considered as it is done by other slip detection methods. Instead of that a slip reduction technique has been developed using measured forces at the wheel contact point [Marx2009]. In a physical sense a loss of traction is the result of too low friction force compared to affecting lateral and tangential forces at the contact point between wall and wheel (see section 3.4.5). In this case, equation (3.5) is violated. As mentioned before, each drive unit of CROMSCI is equipped with a load cell to measure forces at the wheel contact point. The maximum transferable force in rolling direction $\hat{F}_{W_1|x}^{\text{max}}$ can now be estimated according to equation (6.9) based on an estimated static friction value $\hat{\mu}_{\text{stat}}$, the current downforce $F_{W_1|z}^{\text{act}}$ and the sideward force $F_{W_1|y}^{\text{act}}$:

$$
\hat{F}_{W_1|x}^{\text{max}} = \sqrt{\left(\hat{\mu}_{\text{stat}} \cdot F_{W_1|z}^{\text{act}}\right)^2 - \left(F_{W_1|y}^{\text{act}}\right)^2}
$$

(6.9)
6.2. Functionality and Integration

The traction control system adjusts the wheel propulsion, if the current force in driving direction reaches a specific percentage of this maximum. The procedure of traction control is divided into two steps resulting from the velocity controller on the DSP, which uses a proportional and an integral proportion, and is able to change the motor velocity by adjusting the pulse-width modulation (PWM) value. Traction control in combination with a position controller would be also possible, but is not needed here. Both steps of the traction control system have to fit the overall control structure as shown in figure 6.5. Descriptions of the controller and hardware symbols can be found in table A.3 in appendix A.2.

At first, the current PWM limit \( I_{W_i|v}^{\text{max}} \) of wheel \( i \) is trimmed according to equation (6.10). The former maximum \( I_{W_i|v}^{\text{max}'} \) is enhanced by update factors with \( I_p \ll I_m \) depending on the comparison of current \( F_{W_i|x}^{\text{act}} \) and maximal transferable force \( F_{W_i|x}^{\text{max}} \) in rolling direction. These update factors allow a quick limitation and a much slower regeneration of the PWM limit. \( \hat{I}_{W_i|v}^{\text{up}} \) and \( \hat{I}_{W_i|v}^{\text{up}} \) are lower and upper limits of the PWM value to assure a minimum of energy in the one, and to prevent an overstraining in the other case.

\[
I_{W_i|v}^{\text{max}} = \begin{cases} 
I_{W_i|v}^{\text{max}'} - I_m, & \text{if } F_{W_i|x}^{\text{act}} > F_{W_i|x}^{\text{max}} \land I_{W_i|v}^{\text{max}'} > \hat{I}_{W_i|v}^{\text{up}} \\
I_{W_i|v}^{\text{max}'} + I_p, & \text{if } F_{W_i|x}^{\text{act}} < F_{W_i|x}^{\text{max}} \land I_{W_i|v}^{\text{max}'} < \hat{I}_{W_i|v}^{\text{low}} \\
I_{W_i|v}^{\text{max}'}, & \text{else}
\end{cases} 
(6.10)
\]

The traction control system limits the final PWM maximum \( I_{W_i|v}^{\text{max}} \) to specific boundaries to ensure a minimum of force and to avoid a complete stop down of the robot. The chosen update factors allow the reduction of the PWM limit from the maximum to the minimum value within a short period of time and a much slower recovery. The PWM value for the steering motor \( I_{W_i|x}^{\text{act}} \) is not adapted while the PWM value for the locomotion \( I_{W_i|v}^{\text{act}} \) is calculated as given in equation (6.11):

\[
I_{W_i|v} = \begin{cases} 
I_{W_i|v}^{\text{max}}', & \text{if } I_{W_i|v} > I_{W_i|v}^{\text{max}} \\
-I_{W_i|v}^{\text{max}}, & \text{if } I_{W_i|v} < -I_{W_i|v}^{\text{max}} \\
I_{W_i|v}, & \text{else}
\end{cases} 
(6.11)
\]

Unfortunately, the adjusted PWM maximum \( I_{W_i|v}^{\text{max}} \) will lead to an increasing integral value \( I_{W_i|v,I} \) of the PI motor velocity controller. This will cause an increasing controller value again, which is counterproductive. Therefore, the integral limit of the velocity controller has to be adapted within the second step regarding equation (6.12). By this calculation the absolute sum of proportional \( I_{W_i|v,P} \) and integral value \( I_{W_i|v,I} \) cannot be larger than the PWM maximum. The final parameters for the traction control are given in table A.4.

\[
I_{W_i|v,I} = \begin{cases} 
I_{W_i|v}^{\text{max}} - I_{W_i|v,P}, & \text{if } \left( I_{W_i|v,P} + I_{W_i|v,I} \right) > I_{W_i|v}^{\text{max}} \\
-I_{W_i|v}^{\text{max}} + I_{W_i|v,P}, & \text{if } \left( I_{W_i|v,P} + I_{W_i|v,I} \right) < -I_{W_i|v}^{\text{max}} \\
I_{W_i|v,I}, & \text{else}
\end{cases} 
(6.12)
\]
Figure 6.5: Closed-loop velocity controller with integrated traction control system. The force values from the load cell are used to adapt PWM and integral limits.

6.2.2 Experimental Results and Benefit for Safety

The experimental results show the operability, but also the limitation of traction control. Figure 6.6 depicts the effect of immediate acceleration, if the real robot CROMSCI moves upwards a wall: If the traction control system is deactivated (green graph), the wheel slip constrains the motion of the robot and the system is only able to reach a height of 0.028 m. In the other case (blue graph), the power of the drive motor is adapted and the robot is able to move up to eight times higher and reaches a height of 0.20 m. Nevertheless, a general slippage as mentioned before cannot be avoided via TCS since the system has to deal with high forces e.g. based on gravity and friction of the sealings. These contrary forces have to be overcome by the locomotion system and produce a high general slippage during continuous movement. Therefore, the driven distance calculated by vehicle odometry (gray graph) is much higher than the measured distance. Without slip, e.g. on a horizontal plane, the robot would have driven a distance of nearly 1.1 m.

Of course, the most wheel slip occurs if the robot goes up the wall at right angle. But, beside the individual motion direction wheel slip also depends strongly on further environmental aspects, like the character (e.g. shape or roughness) of the surface. That is why the worst case has been examined and described here. While driving downwards, the wheel slip is neglectable since the wheels just have to let the robot down. In the case of driving parallel to the ground, there occurs a sideward slip due to squeezing of the wheel...
6.2. Traction Control System

rubber, which can only be handled by an adaption of the direction of motion, but not by further control elements. But, although there is still a large gap between odometry and controlled motion, the benefit of the TCS (as given in blue graph in figure 6.6) is obvious compared to the case without additional controlling (green graph).

Figure 6.6: The driven vertical distance with and without traction control system (TCS).

According to this, figure 6.7 depicts the pulse width modulation (PWM) values of the left drive motor\(^2\) during this experiment. Again, the two cases of active and inactive TCS are depicted and the graphs show the three motion phases of the robot starting at seconds 2, 5 and 8. As it can be seen, the uncontrolled drive (green) always reaches the PWM maximum of 20,000 whereas the drive with activated TCS lies between a value of 16,000 and 17,000. It has to be considered, that this traction control enables the robot to move further with less current. The PWM value of about 4500 between the motion phases is needed to keep the robot at that position. Otherwise, the robot would roll down again.

Figure 6.7: Pulse width modulation values both for inactive (green) and active TCS (blue) during the experiments shown in figure 6.6, which show the three motion phases of the robot.

\(^2\)Here the comparison of both cases is clearer than of the other two wheels, but the results are similar to the front and right wheel.
Figure 6.8 shows the ratio of sideward forces $F_{W1\,xy}$ (compare equation (3.5)) to the absolute downforce $F_{W1\,z}$ of the front wheel. This ratio is very interesting due to the fact that it allows a conclusion of a dangerous slipping of the robot. Therefore, equations (3.4) and (3.5) can be transformed to the following inequality for a specific wheel $W_i$ (equation (6.13)):

$$
\mu_{stat} \geq \frac{F_{W1\,xy}}{F_{W1\,z}} = \frac{\sqrt{(F_{W1\,x})^2 + (F_{W1\,y})^2}}{F_{W1\,z}}
$$

(6.13)

This ratio describes how large the static friction value must be at minimum to be able to transfer the sideward forces. If this inequality is not fulfilled, the wheel will start slipping. In the example from figure 6.8, the ratio was enhanced by the TCS, but stayed below the used estimated friction coefficient $\hat{\mu}_{stat} = 0.8$ in both cases. Nevertheless, this requirement is not guaranteed as shown in the next section (figure 6.12).

![Figure 6.8: Ratio of sideward forces to downforce of front wheel, which has been improved from an average value of 0.56 during the three motion phases to 0.52 with activated TCS.](image)

The advantages of the presented traction control technique for robot safety compared to a system without TCS can be summed up as follows:

- Increased lifetime of wheels due to less wheel slip,
- increased lifetime of drive motors due to lower permanent current,
- better transfer of holding and motion forces, lower chance of robot slip,
- and reduced time in operation because of faster robot motion.

Nevertheless, it is obvious, that the traction control system cannot annul physics and that a general wheel slip during continuous motion cannot be avoided. It has to be kept in mind that even cars have a wheel slip of 10 to 20% on dry asphalt – and they do not overcome gravity.
6.3 Shear Force Controlling

As shown in equation (6.9), the maximum transferable force in rolling direction \( F_{\text{max}}^{W_i | x} \) of a wheel is reduced by arising sideward forces \( F_{\text{act}}^{W_i | y} \) in y-direction of the wheel. These unmeant forces can occur because of steering transitions as shown in figure 2.5, incorrect initializing or runtime errors of the turning wheel domes, and should be avoided as pronounced in requirement 8. Besides the greater risk of slip, these shear forces also have a negative effect on the wheel’s abrasion and on the mechanical structure of the robot. Therefore, an additional control system for the wheel steering is needed to minimize contrarily orientations in order to reduce sideward forces as demanded by requirement 8.

6.3.1 Functionality and Integration

To decrease the shear forces as illustrated in figure 3.12, the current force values of each wheel \( F_{\text{act}}^{W_i | z} \) and its orientation \( \varphi_{W_i} \) are needed. Because of a missing inter-dsp-communication this has to be done on a pc running with a short cycle time. The first step is to calculate desired forces for each wheel, depending on the percentage of current downforce \( F_{\text{act}}^{W_i | z} \) compared to total downforce \( F_{\text{act}}^{D | z} = \sum_j F_{\text{act}}^{W_j | z} \) of the drive system with \( j \in \{1, 2, 3\} \), as given in equation (6.14).

\[
\begin{pmatrix}
F_{\text{des}}^{W_i | x} \\
F_{\text{des}}^{W_i | y}
\end{pmatrix}
= \frac{F_{\text{act}}^{W_i | z}}{F_{\text{act}}^{D | z}} \cdot R(-\varphi_{W_i}^{\text{act}}) \cdot \sum_j R(\varphi_{W_j}^{\text{act}}) \cdot \begin{pmatrix}
F_{\text{act}}^{W_i | x} \\
F_{\text{act}}^{W_i | y}
\end{pmatrix}
\]  

(6.14)

The forces in y-direction can now be reduced by adapting the steering angles. A pt-similar controller with amplification parameters \( K_{\text{SFC}|\varphi,P} \), \( K_{\text{SFC}|\varphi,I} \) and an initial downforce \( F_{R|z}^{\text{init}} \) which is one third of the robot’s weight, are used to calculate new steering values. The desired values \( \varphi_{W_i} \) calculated by the kinematic control system and the drive conformation module are adapted by a proportional and an integral component as shown in equation (6.15). Here, \( I_{\text{SFC}|\varphi_{i}}(t) \) is the integral value at discrete time \( t \) which is the sum of all past error signals:

\[
\varphi_{W_i}^{\text{offset}}(t) = K_{\text{SFC}|\varphi,P} \cdot \frac{F_{R|z}^{\text{init}}}{F_{\text{act}}^{W_i | z}} \cdot \left( F_{\text{des}}^{W_i | y}(t) - F_{\text{act}}^{W_i | y}(t) \right) + K_{\text{SFC}|\varphi,I} \cdot \frac{F_{R|z}^{\text{init}}}{F_{\text{act}}^{W_i | z}} \cdot \sum_{\tau=1}^{t} \left( F_{\text{des}}^{W_i | y}(\tau) - F_{\text{act}}^{W_i | y}(\tau) \right) = I_{\text{SFC}|\varphi_{i}}(t) \}
\]

(6.15)

The same counts for forces in rolling direction. Due to slightly different turning speeds it is possible that one wheel lies a bit ahead of its desired contact point. In this case, a force in its x-direction is created, which affects the other two wheels in a contradictive way. These forces can be reduced in the same way by calculating velocities offsets \( v_{W_i}^{\text{offset}} \) with parameters \( K_{\text{src}|\varphi,P} \) and \( K_{\text{src}|\varphi,I} \) and by comparing forces in rolling instead of sideward direction as given in equation (6.16). The general difference in adjusting steering position and wheel velocity is, that the steering offset has to be added continuously while the
velocity offset only has to compensate present straight shear forces. The reason is based on the nature of things: Steering offsets have to be kept up to avoid a return of these unwanted forces.

\[ \vec{v}_{R_W} = \vec{v}_{R_W} + \vec{v}_{\text{offset}} \]

Figure 6.9: Simplyfied example for wheel steering adaptions which become necessary e.g. because of a wrong initialization (a). Offset angles are added permanently to balance out unwanted shear forces (b).

Figure 6.9 gives an example for that: Here the shear force controller is used to set the wheels parallel while the robot drives in a straight line. It is obvious, that the offsets have to remain also in the case of a change in the driving direction. In contrast to that the wheel velocity offset has to be set only depending on the current state because this adds a permanent position offset to the wheel in relation to the other wheels. Therefore, the integral parameter is set to \( K_{SFC|v,1} = 0 \), which reduces the controller to a simple P-controller.

\[
v_{W_i}^{\text{offset}}(t) = K_{SFC|v,P} \cdot F_{R_W}^{\text{init}} \cdot \left( F_{W_i|x}^{\text{des}}(t) - F_{W_i|x}^{\text{act}}(t) \right) \text{ proportional portion} \\
+ K_{SFC|v,I} \cdot \sum_{\tau=1}^{t} \left( F_{W_i|x}^{\text{des}}(\tau) - F_{W_i|x}^{\text{act}}(\tau) \right) \text{ integral portion} \] (6.16)

The final steering and velocity values \( \varphi_{W_i}^{\text{final}} \) and \( v_{W_i}^{\text{final}} \) are given in equation (6.17) and simply add the offset values to the desired commands from the kinematic calculation. Table A.4 gives an overview on the controller parameters, which have finally been used.

\[
\varphi_{W_i}^{\text{final}} = \varphi_{W_i} + \varphi_{W_i}^{\text{offset}}, \quad v_{W_i}^{\text{final}} = v_{W_i} + v_{W_i}^{\text{offset}} \] (6.17)
6.3.2 Experimental Results and Benefit for Safety

The first experiments of the shear force controller on the real prototype CROMSCI have been executed on flat ground. This setup has been chosen to exclude additional sideward forces, e.g. based on gravity, which affect the measurements. Figure 6.10 shows the lateral force values of all three wheels with SFC (blue) and without the controller (green). During both experiments the real robot executed the same trajectory with strong changes in motion and turning. In table 6.2, the average values of the absolute lateral forces are given. In average, these shear forces are reduced by 63.2% via the introduced SFC.

![Figure 6.10: Lateral forces $F_{W_{iy}}$ of all three wheels during robot motion on the ground. The force values with active SFC (blue graphs) are lower than without the controller (green graph).](image)

<table>
<thead>
<tr>
<th>$F_{W_{iy}}$</th>
<th>without SFC</th>
<th>with SFC</th>
<th>reduction by</th>
</tr>
</thead>
<tbody>
<tr>
<td>front wheel</td>
<td>10.56 N</td>
<td>3.06 N</td>
<td>71.0%</td>
</tr>
<tr>
<td>left wheel</td>
<td>5.80 N</td>
<td>2.62 N</td>
<td>54.8%</td>
</tr>
<tr>
<td>right wheel</td>
<td>6.21 N</td>
<td>2.63 N</td>
<td>57.6%</td>
</tr>
<tr>
<td>average</td>
<td>7.52 N</td>
<td>2.77 N</td>
<td>63.2%</td>
</tr>
</tbody>
</table>

These results show the general operability of this controller. But, in cases of vertical motion, additional sideward forces based on gravity occur, e.g. if the robot drives horizontal and all holding forces are in wheel’s y-direction. In this case the unwanted portions of
sideward forces are those which differ from the average. Equation 6.18 shows the calculation of this difference value $\Delta F_{W_i|y}^{avg}$. It does not use the simple average of lateral forces due to the fact that wheels with a higher downforce $F_{W_i|z}$ will also generate higher sideward forces. Therefore, the division factor of the average is the ratio of wheel’s downforce divided by the sum of downforces instead of three:

$$\Delta F_{W_i|y}^{avg} = \frac{\text{weighted average}}{\sum_j F_{W_j|z}^{act} - F_{W_i|y}^{act}}$$  \hspace{1cm} (6.18)

The results of this calculation are plotted as graphs in figure 6.11 which shows two experiments with the robot CROMSCI attached to a wall, one with shear force controlling and one without. Unfortunately, these results are disturbed by electromagnetic pulse effects, which influence the loadcell measurements at two periods of time between seconds 10-12 and 20-27. Nevertheless, the controller as well as the robot itself were able to execute their tasks without greater problems.

The average values of all six graphs are given in table 6.3, which shows the benefit of this mechanism: The undesired shear forces in y-direction are decreased by 13 N in average, which is a reduction of one third.

![Graph showing differences of current lateral force values and the average force.](image-url)
Table 6.3: Average force differences according to figure 6.11 as calculated by equation 6.18.

<table>
<thead>
<tr>
<th>without SFC</th>
<th>with SFC</th>
<th>reduction by</th>
</tr>
</thead>
<tbody>
<tr>
<td>front wheel</td>
<td>44.93 N</td>
<td>18.17 N</td>
</tr>
<tr>
<td>left wheel</td>
<td>34.66 N</td>
<td>32.63 N</td>
</tr>
<tr>
<td>right wheel</td>
<td>39.39 N</td>
<td>28.27 N</td>
</tr>
<tr>
<td>average</td>
<td>39.66 N</td>
<td>26.35 N</td>
</tr>
</tbody>
</table>

The absolute values of lateral forces, as depicted for the front wheel in figure 6.12, show, that the overall forces are lower if the SFC is active. In absolute terms one can state that the force $F_{W_{1y}}^{act}$ has been decreased from 64.32 N to 41.74 N in average which corresponds to a reduction of 35 %. The sideward forces still reach values of up to ±150 N, but these values are the required holding force of the robot weight. The absolute force values of the other two wheels have not been improved that much, but the forces are distributed much more uniform in general.

Figure 6.12: The lateral forces of the front wheel while moving up a wall with permanently changing velocity and direction commands - with and without shear force control (top). Below, the ratio of sideward forces to downforces is shown.

Figure 6.12 also shows the ratio of sideward forces $F_{W_{1xy}}$ to the absolute downforce $F_{W_{1z}}$ of the front wheel, as already used in case of the TCS (see equation (6.13)). This ratio is used as a dimension of the chance of wheel and robot slipping. The average ratio of the plot given in figure 6.12 is reduced from 0.543 to 0.429 by the SFC. Even better is the result, that this ratio remains below the estimated friction value $\hat{\mu}_{stat}$, which has
been determined experimentally\textsuperscript{3} on flat ground. As shown in this figure, the green graph exceeds the estimated friction value two times. The first time at 22 s, the peak seems to be caused by electromagnetic disturbances, which influence load cell measurements. But, at time 29 s, the ratio reaches the friction value resulting in a great chance of wheel slip. Of course, the real friction value can be at 0.85 so the wheel would not slip in this situation, on the other hand it can also be at 0.7 so the wheels would slip more often. This experiment shows the great importance of this measure to improve robot adhesion. If the other wheels are affected in a similar way as given in figure 6.12, the whole robot would slip, which could lead to a complete drop-off of the system.

These results are much more impressive than those from the experiment shown in figure 6.8. The reason lies within the driven trajectories: In the first experiments regarding the TCS the robot moved forward in a straight line so only neglectable lateral (shear) forces occurred. The trajectory of the second experiment was more varied with many motion changes, which produced much more shear forces. The advantages of the SFC for robot safety can be summed up as follows:

- Complete avoidance of wheel and robot slip in certain situations,
- lower chance of wheel and robot slip in general,
- and increased lifetime of wheel rubber and wheel mechanics due to lower sideward forces.

### 6.4 Setup of the Advanced Motion Control Components

Figure 6.13 shows the exact setup and interaction of the safety-relevant shear force and traction controllers. The desired steering and driving commands $\varphi_{W_i}$ and $v_{W_i}$ from the kinematics are adapted by the SFC (yellow) based on current downforce values. Each DSP circuit board with motor controllers, electronics, and amplifiers is responsible for one drive unit and generates PWM signals for steer and drive motors. Based on the estimated friction value $\hat{\mu}_{stat}$ and current forces, the PWM value of the motor for locomotion is limited to $I_{max}^{W_i}$ by the TCS (red), which is an integrated component of the DSP program. This advanced motion control system runs parallel to the force and negative pressure control system and has no influence on the adhesion force of the robot, as it will be described in the next chapter 7. Nevertheless, TCS, SFC, and the boundary conditions are important for robot safety in terms of the avoidance of robot slip by enhanced friction.

The final control parameters and limits are presented in appendix table A.4. Most of these parameters have been determined via experiments with the real robot CROMSCI. The update factors $I_p$ and $I_m$ (compare equation (6.10) in section 6.2.1) e.g. are not highly safety critical since a malfunction of this controller based on wrong parameters does not lead to a drop-off. They only describe the speed of engagement and recovery of the TCS. But of course, the traction control will not work correctly if these values do not fit the entire system. In the worst case, the TCS will not become active if e.g. $I_m$ is too

\textsuperscript{3}The traction control system uses a friction value of 0.8.
small or \( I_p \) is too large. The same counts for thresholds \( \dot{I}_{W_1|v}^{\text{low}} \) and \( \dot{I}_{W_1|v}^{\text{up}} \), which define the minimum and maximum PWM values. Depending on the lower limit, the motor current might be reduced to zero which would prohibit robot movement completely. Otherwise it is also possible that the TCS will not be able to reduce the current to a suitable value, if the threshold is chosen too high. As a result, the wheels would slip more than desired. In contrast to this the parameters of the SFC have to be chosen more carefully. Since it influences actively wheel steering and turning, an undesired oscillation might be possible if the control parameters \( K_{SFC} \) are set wrong (compare equations (6.15) and (6.16) in section 6.3.1). In the worst case it could lead to a drop-off of the robot, if the SFC produces such heavy overswinging or build up until the wheel domes rotate on the spot and cause robot slip. The experiments have shown, that the given parameters are suitable for the given tasks and that the overall system safety could be improved by the described measures.

After handling aspects of locomotion and the corresponding safety requirements from section 5.2.2, the next chapter will deal with the adhesion system of the robot and present the behavior-based network which is responsible for the closed-loop adhesion control.
7. Behavior-Based Adhesion Control

Uncertainty is one of the most severe problems in robotics. One can neither be certain about the sensor values nor about their correct interpretation. In the present case of climbing robots, there exists an uncertainty about the influence of the surface structure on the adhesion system. This makes it necessary to develop a robust control system and additional components, which ensure safety. This includes those elements as presented in figure 5.8 reaching from closed-loop controllers up to higher control strategies.

This chapter focuses on the adhesion controllers, which are most important for robot safety. Section 7.1 introduces the general concept of the developed behavior-based adhesion control network and gives an overview on the individual components. The following sections 7.2 to 7.4 will present the single elements of the behavior-based control system more detailed. These descriptions are separated into three parts which summarize associated control elements: The basic chamber controllers, chamber deactivation elements and force balancing methods. Their individual contributions to robot safety will be examined in-depth. Finally, the interaction of the complete behavior-based network will be given in the last section 7.5. The focus lies here on the correct operation of all individual components of the adhesion system to achieve a complex and robust system behavior to enhance robot safety. The motion control measures will not be taken into account here, since they work separately from the other methods and do not influence them.

7.1 Behavioral Concept for Adhesion Control

As pointed out before, the existing components of the adhesion control system, as shown in section 2.1.3, have been transferred and integrated into the behavior-based ib2C framework, which has been introduced in section 2.2.1. This has different advantages:

**Meta data** The behaviors deliver meta data, which abstract from specific settings like pressure and force values, velocities or positions. Each behavior has an onboard rating mechanism, which points out how satisfied a behavior is and how strong its influence should be in a specific situation. This makes the safety analysis independent from vehicle parameters, if the so-called target rating and activity values are chosen properly for each behavior.
Extensibility The behavior network can be created bottom-up and individual elements can be added and removed easier than in regular systems. The integration of new components can be done without disturbing the existing elements. Of course, certain adaptations e.g. in terms of inhibitions or stimulations have to be done, but the overall labor costs are lower.

Analysis Because of the homogeneous behavior interface, the online and offline analysis of the network is much easier since a couple of visualization and analysis tools exists, which help to identify undesired characteristics. These possibilities are important for the fulfillment of requirement 4, which demands an online identification of problems in the control components.

The developed adhesion control system contains behaviors, which can be split up into an inner part containing behaviors for pressure control and an outer part for force control. Each behavior is responsible for one task, function or component, and follows the design principles of the iB2C architecture (compare section 2.2.1). Therefore, each component provides meta data showing its activity and target rating. This uniform interface is the strength of the behavior-based control architecture and allows an easy interaction of the individual components as well as an intuitive interpretation of their states. The remaining question is, in which way the different pressure control behaviors are able to interact and how the control system looks like. This section will deal with these aspects and starts with an overview on the complete adhesion control system with the individual components. The upcoming sections will describe the single behavioral components more detailed.

7.1.1 The Adhesion Control System

The adhesion control system is the most important component for robot safety. Therefore, its operability and reliability have to be ensured during all robot operations. The idea is to split the existing controllers into individual and stand-alone units. Via its meta data and further control values these elements can interact, if it is necessary. Beside the general advantages of a behavior-based network as mentioned before, the new adhesion control network should fix problems of the former adhesion controllers and enhance their functionalities concerning operability, safety and handling:

Fast calculation of pressure distribution The pressure calculation, which computes individual chamber pressures based on a desired total force, uses an iterative algorithm. This may cause problems regarding the execution time of the controllers and will be replaced by a combination of direct and iterative calculation as given in section 7.4.4.

Consideration of chamber limits Another serious problem of the existing pressure calculation module is, that it does not consider chamber limits. Only inactive chambers are taken into account. But, in practice it is not possible for some chambers to reach the desired pressure because of leakages to the ambient air and to neighbor chambers. As a result, the negative pressure system will not be able to adjust the desired downforce.
More reliable deactivation of chambers  The leakage estimation is a suitable way to identify leak chambers. Unfortunately, there still remain situations in which the estimation fails. An additional value is the pressure trend, which tries to predict if the chamber will lose negative pressure within the next period of time.

Faster reintegration of inactive chambers  To reduce the stress of the adhesion system if several chambers have been deactivated, the reintegration process has to be speeded up. Therefore, the chamber deactivation will be updated as shown in section 7.3.

More intuitive analysis  Regarding the classic controllers, it was difficult to identify reasons for system malfunctions or undesired reactions and controller problems (compare requirement 4). The new control system with individual behavior elements allows a more detailed view and an easier online-analysis than before.

The transformation of the existing control structure to the behavior-based network has great impact on the software components. The behavior-based pressure control system is the lowest part of the adhesion system and therefore has to be set up first. It contains the individual behaviors to close-loop control each chamber pressure separately, as shown in figure 7.1. The most important aspect is, that each chamber has its own pressure control behavior and deactivation element. The general principle remains the same: The chamber pressures are adjusted within an inner control loop consisting of pressure controller, sensors and valves. An outer control loop uses the negative pressure control system as an actuator to balance out robot downforce and its point of downforce.

The main question that arises at this point is, why the existing controllers were not sufficient. In principle, the individual behaviors are still closed-loop controllers, which perform similar tasks as before. The same functionalities could also be implemented as classic cascaded controllers – but this counts only for the range of functions concerning the adhesion control. This does not count for the advanced analysis features of iB2C networks and especially for the superimposed risk prediction component, as presented in the following chapter 8. The usage of behaviors with a common interface for interaction allows it to develop an approach for risk prediction, which works for behavior-based networks in general. Therefore, the use of behaviors is the best way to combine classic control methods with additional functions. Therefore, the closed-loop control components have been realized in form of a behavior-based network with individual, but interacting elements, as depicted in figure 7.1. The individual behaviors and preprocessing calculation modules will be presented in the next sections.

7.1.2 Behaviors and Modules

As shown in figure 7.1, the adhesion control system consists of a couple of behaviors and other computational units. Depending on the desired characteristics of the network, some values like $F_{N|z}$ can either be set manually by the user or via other behaviors. In the context of safety and risk analysis it is important to know, which behaviors are necessary for safe adhesion of the robot, how they work, and which additional meta data they generate. The following list will give a short overview on the tasks and functionalities of the behaviors. Each of the components will be explained in-depth later in this chapter.
7. Behavior-Based Adhesion Control

Figure 7.1: Concept of behavior-based adhesion control structure including the data flow (some control connections are hidden which will be presented in the next sections).

Upcoming figure 7.2 gives a more detailed view on the three **basic control elements** (green modules) introduced in figure 7.1, which are responsible for chamber and reservoir control:

**Chamber control** This behavior is the central control element and responsible for **basic chamber pressure control** of one single chamber (section 7.2.1). It exists eight times within the network and tries to generate the desired amount of negative pressure by adjusting the valve opening. Seven instances are located inside of the chamber control group, whereas the eighth behavior controls the reservoir pressure (chamber 8).

**Chamber control group** The control group contains the individual chamber control behaviors, which are responsible for the seven adhesion chambers. It generates meta data to depict the state of the complete negative pressure system depending on the individual elements.

**Reservoir Control** This component is responsible for controlling the corresponding chamber behavior (chamber 8) and for the calculation of meta data, which express the state of the reservoir pressure related to the seven adhesion chambers.
As introduced before, a controlled shutdown of chambers is helpful in cases of high leakages. Therefore, some additional components are necessary to calculate important state information and to deactivate a chamber. Figure 7.4 will present the elements of the chamber deactivator group (blue module) more detailed later on:

**Chamber deactivator group** This group is responsible for the deactivation of the adhesion chambers in cases of high leakages and contains seven individual groups – one for each working chamber. Each of these groups consists of a chamber deactivation behavior and associated behaviors for leakage detection and pressure trend, which signal upcoming problems of that specific chamber.

**Chamber deactivation** The deactivation behavior as described in section 7.3.3 is responsible for the deactivation, but also for the testing and reintegration of one single chamber. It is embedded into the chamber control group as well as the next two behaviors.

**Chamber leakage** This behavior is used to estimate the current leakage of a chamber and triggers the corresponding chamber deactivation behavior if the leakage is above a certain threshold.

**Chamber pressure trend** Similar to the previous leakage behavior, this behavior triggers a chamber deactivation if it is necessary. To achieve this, it estimates, how long this chamber will keep a certain amount of negative pressure.

Of course, it is not very practical to set desired negative pressures for all chambers separately by hand, as it is possible so far. Therefore, the behavioral network is enhanced by components to control a global downforce (red modules in figure 7.1) and to calculate individual chamber pressures depending on this force (yellow modules). Additionally, some computational modules – which are no behaviors – are used, which preprocess the sensor data and calculate desired values. All these elements will be illustrated in figure 7.12.

**Force controller** This behavior group is responsible for force control and makes use of the negative pressure system as a kind of actuator, as described in section 7.4. Its task is to balance out the downforces of the wheels for an uniform distribution and to adjust a certain total amount of adhesion force.

**Force point calculation** This module calculates the current downforce value and point of downforce depending on the forces at the wheels measured by the load cells. The same is done for the negative pressure system depending on the pressure values and suction areas.

**Chamber pressure calculation** The task of this behavior is the calculation of individual chamber pressures depending on a desired total downforce (see section 7.4.4). It takes the robot geometry and present limits of the negative pressure system into account to set desired pressure values.

**Min/max pressures** This element is responsible for calculating chamber pressure limits depending on their current state. This information is needed to adapt the adhesion system, if one or more chambers have reached their limits.
All these elements will be described more detailed concerning their functionalities and behavioral information in the following paragraphs. This description is given bottom-up similar to the development procedure starting with the basic chamber pressure controller and reaching up to the force controller on top.

7.2 Chamber and Reservoir Control

The most important components of the adhesion system are the closed-loop pressure controllers. This control system has been setup in a way that it is a combination of eight individual control behaviors: Seven for the working chambers and one for the reservoir. The seven single chamber control behaviors are arranged into a chamber control group as shown in figure 7.2. Since the reservoir chamber does not contribute to the adhesion directly, its control behaviors are located outside of this group.

![Figure 7.2: Overview on the basic chamber control network which is part of the complete network given in figure 7.1.](image)

7.2.1 Basic Chamber Control Behavior

The elementary component is the chamber control behavior. Each of these – in total eight – behaviors performs the closed-loop controlling of its corresponding chamber, which can be of different types depending on the selected control mode $c_{CC_i} \in \{c_{off}, c_{init}, c_{pos}, c_{pres}, c_{flow}\}$.

**Valve position control** If this controller is selected ($c_{CC_i} = c_{pos}$), the behavior just forwards the desired valve area $A_{V_i}$ from user input to the output. It is commonly used for debugging purposes only. Only the reservoir chamber $C_8$ is commonly used to work in this mode with a fully closed valve for a maximal reservoir pressure.

**Chamber pressure control** In this mode ($c_{CC_i} = c_{pres}$), the current chamber pressure $p_{C_i}$ is adjusted by the behavior as described in the next paragraph. It is one of the two control modes which can be used in operation for the working chambers.

**Air flow control** This control mode ($c_{CC_i} = c_{flow}$) is the second one, which can be used for controlled robot adhesion. In contrast to the previous mode, it tries to balance the air flow between reservoir and working chamber.
Two additional modes are needed, but do not perform any controlling: $c_{CC}^{off}$ deactivates the controller, which is the default case on startup. The initializing procedure of a chamber valve is triggered by $c_{CC}^{init}$ and needed to set up the stepper motor counter. This is done by opening the valve to the maximum until an internal switch is reached, which resets the step counter.

**Chamber Pressure Controller**

Because of the high dynamic of the control process, a simple $P$-controller is not sufficient. An integral portion becomes necessary because of variable process parameters like sealing leakages and the reservoir pressure. A differential portion is needed to react on fast pressure changes. Therefore, a classic PID-controller is applied as given in equation (7.1) with error difference $\Delta p_{Ci} = p_{Ci}^{des} - p_{Ci}^{act}$, PID parameters (amplification $K_{CC|p}$, reset time $T_{CC|i}$ and derivative time $T_{CC|d}$) and sampling time $\Delta t$. As output, the controller delivers the desired valve area $A_{Vi}$ to regulate the airflow between chamber and reservoir.

$$
A_{Vi}(t) = K_{CC|p} \cdot \Delta p_{Ci}(t) + K_{CC|i} \cdot \frac{\Delta t}{T_{CC|i}} \cdot \sum_{\tau=1}^{t} \frac{\Delta p_{Ci}(\tau) - \Delta p_{Ci}(\tau-1)}{2} + K_{CC|d} \cdot \frac{T_{CC|d}}{\Delta t} \cdot \left(p_{act}(t) - p_{act}(t-1)\right)
$$

Equation (7.1)

Additionally, the controller uses an *anti-reset-windup* system to limit the integration part if the control value – which is the valve’s position – is at the limit of the hardware. In other words: The valve cannot be more than open.

**Closed-loop Control of the Air Flow**

The main problem of the standard pressure controller lies within the nonlinear control path. Changes in the reservoir pressure caused by the other chambers have influence on the massflow between chamber and reservoir and have to be handled as disturbances by the pressure controller. Therefore, the idea is to control the massflow $\dot{m}_{Vi}$ between chamber and reservoir instead of the pressure. In this way, pressure changes of the chamber or the reservoir are transferred directly to a valve area to keep the massflow constant. In the optimal case the resulting airflow is equal to the air lost via leakages $\dot{m}_{Vi} = \dot{m}_{Li}$.

The massflow through valve $i$ can be calculated according to equation (7.2) depending on the chamber pressure $p_{Ci}$, reservoir pressure $p_{C_R}$, valve area $A_{Vi}$ and density of air $\rho_{air} = 1.183$ kg/m$^3$, which is assumed constant as described in [Hillenbrand2009]. Again, an *anti-reset-windup* method is used to limit the integrational part based on real valve limits.

$$
\dot{m}_{Vi} = \text{sgn}(p_{C_R} - p_{Ci}) \cdot A_{Vi} \cdot \sqrt{2 \cdot \rho_{air} \cdot |p_{C_R} - p_{Ci}|}
$$

Equation (7.2)

Of course, both controllers have to meet certain conditions to fulfill their tasks in an optimal manner. Requirement 1 points out that the adhesion controllers must be robust and oscillation-free while balancing disturbances out.
Behavioral Information

The meta data of the chamber controller are calculated depending on the selected control mode \(c_{CC_i} \in \{c_{off}, c_{init}, c_{pos}, c_{pres}, c_{flow}\}\). As given in equation (7.3), the target rating \(r_{CC_i}\) of a chamber control behavior \(i\) depends on chamber parameters like current valve area \(A_{V_i}^{act}\) or chamber pressure \(p_{C_i}^{act}\), if position \(c_{pos}\), pressure \(c_{pres}\) or flow control \(c_{flow}\) is selected. \(\Delta p_{\text{nmax}}^{\text{pp}}, \Delta p_{\text{pmax}}^{\text{pp}} \in \mathbb{R}^+\) are thresholds for negative and positive chamber pressure difference, \(A_{V_i}^{\text{max}}\) is the maximum valve area and \(\Delta A_{\text{max}}^{\text{pp}} \in \mathbb{R}^+\) the parameter for valve area difference. In contrast to the target rating, the activity \(a_{CC_i}\) only depends on the current valve opening \(A_{V_i}^{act}\) and the maximum as given in equation (7.4).

\[
\begin{align*}
    r_{CC_i} &= (1 - \iota_{CC_i}) \cdot \begin{cases} 
    1.0, & \text{if } c_{CC_i} = c_{off} \\
    0.5, & \text{if } c_{CC_i} = c_{init} \\
    \left(\frac{\Delta A_{V_i}^{\text{pp}}}{\Delta A_{\text{max}}^{\text{pp}}}\right)^1, & \text{if } c_{CC_i} = c_{pos} \\
    \langle \text{max}\left(\frac{p_{C_i}^{\text{pp}} - p_{C_i}^{\text{act}}}{\Delta p_{\text{pmax}}^{\text{pp}}}, \frac{p_{C_i}^{\text{act}} - p_{C_i}^{\text{pp}}}{\Delta p_{\text{nmax}}^{\text{pp}}}\right)\rangle^1, & \text{else}
    \end{cases} \\
    a_{CC_i} &= \frac{A_{V_i}^{act}}{A_{V_i}^{\text{max}}} \cdot \iota_{CC_i}
\end{align*}
\]

In this context \(\langle x \rangle^1\) describes a function, which limits the content of the brackets into the range of \([0, 1]\). In general, this limit function can be set up with a lower limit \(l \in \mathbb{R}\) and an upper limit \(u \in \mathbb{R}\) with \(l \leq u\) according to equation (7.5). This becomes necessary, since all meta data \(a, r, \iota, s\) have to be in the range of \([0, 1]\).

\[
\langle a \rangle^1 = \begin{cases} 
    l, & \text{if } a < l \\
    u, & \text{if } a > u \\
    a, & \text{else}
    \end{cases}
\]

In the present setup, the parameter for valve difference \(\Delta A_{V_i}^{\text{max}}\) is set to the maximum valve area of \(A_{V_i}^{\text{max}} = 0.0003\) m\(^2\). The thresholds for pressure differences \(\Delta p_{C_i}^{\text{ppmax}}\) and \(\Delta p_{C_i}^{\text{pmax}}\) are set different for the outer chambers, the center chamber and the reservoir. In software they are set in terms of thresholds for negative and positive force difference generated by the corresponding chamber. The pressure thresholds have to be set depending on the individual suction areas and further aspects to represent meaningful values.

It has to be mentioned, that this behavior is unsatisfied if it is inhibited or not stimulated. This relationship is conflictive to one of the design principles postulated by Proetzsch et al.: “There is no (direct, i. e. inside a behavior) influence of the activation \(\iota\) on \(r\)” [Proetzsch2010, p.4]. In the more detailed description of this principle it is said, that \(r\) only depends on the present input \(\vec{e}\) and the internal state of the behavior. Due to the fact, that the chamber pressure behavior is a closed-loop controller, which desires to be active it is obvious, that the internal state of this behavior for sure also depends on its activation. Therefore, this design principle is bended in a way that this kind of inverse influence is allowed.
7.2. Chamber and Reservoir Control

7.2.2 Chamber Control Group

As already shown in figure 7.2, the seven chamber control behaviors are arranged into a behavioral group. This behavior group again delivers activity and target rating values, which are calculated depending on the meta data of the internal behaviors to provide information about the overall state of the negative pressure system. Since the reservoir chamber is not located inside of this group, only the seven working chambers \( i \in \{1, 2, ... 7\} \) are considered here.

Behavioral Information

As said, this behavioral group should express a representative state of the negative pressure system. Therefore, its target rating \( r_{CC} \) (equation (7.6)) is the average of the largest three products of chamber activity and target rating as given in equation (7.7). The background of this calculation is, that the system is in a critical state if these chambers are dissatisfied (\( r_{CC_i} = 1 \)) and active (\( a_{CC_i} = 1, \) valve full open).

\[
r_{CC} = \sqrt[3]{\frac{\sum_{i=1}^{3} r'_{CC_i}}{3}}, \quad (7.6)
\]

\[
r'_{CC_j} = r_{CC_j} \cdot a_{CC_j} \quad \text{with} \quad r'_{CC_j} \geq r'_{CC_{i+1}} \forall j \in \{1, 2, ... 6\}, \quad (7.7)
\]

The activity \( a_{CC} \) of this group is just the average activity of the seven working chambers, as given in equation (7.8):

\[
a_{CC} = \mu_{CC} \cdot \frac{\sum_{i=1}^{7} a_{CC_i}}{7}, \quad (7.8)
\]

7.2.3 Reservoir Control

Whereas the chamber control group represents the negative pressure system, which is responsible directly for robot adhesion, the reservoir control behavior works in the background. It just forwards the desired values for reservoir pressure or valve area, but calculates important behavioral meta data which are used for determining hazardous situations. The idea is to get a state information about the influence of the working chambers on the reservoir. The basic statement is, that the more negative pressure is taken out of the reservoir, the smaller the pressure difference between working chambers and reservoir will be.

Behavioral Information

This coherence between reservoir and chamber pressures is used to calculate the target rating \( r_{RC} \) of this behavior as shown in equation (7.9). Here, \( \tilde{p}_{RC}^{\min} \) and \( \tilde{p}_{RC}^{\max} \) denote thresholds for the pressure difference between reservoir and working chambers with the
highest negative pressure. It has to be noticed, that a minimum of basic leakages at the sealings is an assumption for this rating to work. If all chambers are (totally) leak-tight, the negative pressure would be the same in all eight chambers. Nevertheless, this will not happen on irregular surfaces like concrete. In contrast to that, the activity simply depends on the internal activation \( t_{RC} \) and the selected controller type \( c_{CC} \) (equation (7.10)).

\[
\begin{align*}
r_{RC} &= \begin{cases} 
1.0 & \text{if } c_{CC} = c_{CC}^{off} \\
0.5 & \text{if } c_{CC} = c_{CC}^{init} \\
\frac{p_{RC}^{max} - \min\{p_{C_1}^{act}, ..., p_{C_8}^{act}\}}{p_{RC}^{min} - p_{RC}^{act}} & \text{else}
\end{cases} \\
a_{RC} &= t_{RC} \cdot \begin{cases} 
0.0 & \text{if } c_{CC} = c_{CC}^{off} \\
1.0 & \text{else}
\end{cases}
\end{align*}
\] (7.9) (7.10)

7.2.4 Experimental Results and Benefit for Safety

Compared to the previous classic controller setup, there is no measurable benefit regarding the closed-loop controlling of the individual chambers itself. The same counts for the two different chamber controllers (flow and pressure), which act with the used PID control parameters very similar as shown in figure 7.3. In this experiment inside of the simulation environment, the robot was driven over a deep crack with only the chamber control behaviors activated, once in pressure (green) and once in flow control mode (blue). The figure presents pressure and valve values of the frontal chamber \( C_1 \), the rear chamber \( C_4 \) and the reservoir pressure \( p_{RC}^{act} \). As it can be seen, the resulting pressure \( p_{C_1/4}^{act} \) and valve values \( A_{V_{1/4}}^{act} \) are nearly the same for the considered chambers. This example also shows, that the existing approach as well as the new behavior-based closed-loop control is not sufficient for robot safety.

At the beginning, the robot is driven on a very smooth surface with low basic leakages. At about \( t = 3.5 \) s the frontal chamber reaches a structured patch. In the following, the chamber controllers have to perform more pressure balancing and the reservoir pressure rises slightly. Then, the first chamber \( C_1 \) reaches the crack (\( t = 11 \) s) and its pressure increases to ambient pressure very fast. From now on, the controllers try to adjust the pressure by opening the valves, which results in a loss of reservoir pressure. The increasing reservoir pressure also influences those chambers, which are not affected by the crack directly. This is visualized by the pressure value \( p_{C_4}^{act} \) of the rear chamber, which also starts to increase up to a value close to ambient pressure. The valves of all chambers open to the maximum. In this case the robot has no chance to maintain the desired adhesion forces and drops off, because the adhesion system reaches its limits (valves full opened, suction engines at maximum). The hardware solution would be stronger suction engines and larger valves, which comes with the disadvantages of additional mounting space and a much higher weight, which in turn results in higher needed adhesion and motion forces, higher wheel abrasion and further problems.

The advantage regarding safety does not lie within the closed-loop controlling itself, but within its analysis. Since the network uses a standardized interface for each element, it is possible to get a live-view on the states of the different control elements. This helps to identify problems during runtime and to determine their causes more easily than before.
7.3. Chamber Deactivation

One main threat for climbing robots using negative pressure adhesion is a loss of adhesion force caused by unsealed suction chambers. Of course, it is the task of the closed-loop controllers to balance these variances out to keep the desired amount of downforce in each chamber. But, as shown before, there exist situations, in which the loss of negative pressure cannot be compensated by the negative pressure system. Therefore, one important safety feature of the adhesion control system is the deactivation of those chambers, which do not contribute to adhesion because of high leakages and endanger the whole negative pressure system, as postulated in requirement 13. To achieve this, a set of deactivation behavior groups is used – one for each chamber – to cut off chambers with a high leakage from the negative pressure system (figure 7.4). Triggers for such a shutdown are an estimated leakage value $\hat{A}_L$ and a predicted pressure trend $\Delta \hat{p}_{PT}$ in form of activity values of the corresponding analysis behaviors. The next sections 7.3.1 and 7.3.2 will introduce the analytic behaviors which try to identify leaky chambers. Afterwards, the mechanisms of deactivation and reintegration are presented.

Figure 7.3: Simulation experiment in which the robot tries to overcome a deep crack. The values based on pressure (green) and flow chamber control (green) are nearly congruent. Furthermore, the standardized behavior interface enables an automatic online analysis as presented in chapter 8, which is used to identify upcoming hazards and dangerous system states.

7.3 Chamber Deactivation
7. Behavior-Based Adhesion Control

Figure 7.4: General concept of the deactivation elements and data flow of the used behaviors (compare figure 7.1).

7.3.1 Leakage Estimation Behavior

The leakage estimation behavior is one of the supporting modules for the chamber deactivation behavior as shown in figure 7.4. Its task is to calculate the current leakage value of the corresponding chamber and to signal an appraisal value to the behavior network. The meta values are used by the deactivation behavior to identify a leak chamber.

Estimation of Leakages

For the detection of leak chambers the setup of the negative pressure system and the closed-loop controllers have to be taken into account. The control process can be described as shown in equation (7.11): The current pressure change $\dot{p}_{C_i}$ of a chamber $i \in \{1, 2, ..., 7\}$ can be calculated using the total leakage area $A_{L_i}$, the valve area $A_{V_i}$, outer ambient pressure $p_{amb}$, chamber pressure $p_{C_i}$ and reservoir pressure $p_{C_R}$ as described in [Hillenbrand2009]. The physical constants are density $\rho_{air} = 1.1883 \text{kg/m}^3$ and adiabatic exponent $\kappa_{air} = 1.402$ of air, the ideal gas constant $R = 287.058 \text{J/(kg K)}$, temperature $T_{air} = 293.15 \degree \text{K}$ and the chamber volume $V_{C_i}$.

$$
\dot{p}_{C_i} = \sqrt{2 \cdot \rho_{air} \cdot \kappa_{air} \cdot R \cdot T_{air} \cdot \frac{1}{V_{C_i}}} \left( A_{L_i} \cdot \sqrt{p_{amb} - p_{C_i}} - A_{V_i} \cdot \sqrt{p_{C_i} - p_{C_R}} \right) \quad (7.11)
$$

Assuming, that the chamber pressure will not change during one calculation step – so $\dot{p}_{C_i} = 0$ – equation (7.11) can transformed to calculate the estimated leakage $\hat{A}_{L_i}$ of the given chamber as shown in equation (7.12). This simplification can be done because the dynamical effects like balance streams proceed very fast. It also avoids large noise resulting from the derivation of the pressure value.
\[
\hat{A}_{Li} = \frac{\sqrt{p_{Ci} - p_{CR}}}{\sqrt{p_{amb} - p_{Ci}}} \cdot A_{Vi}
\]  

(7.12)

In the case, that the chamber pressure \( p_{Ci} \) tends towards the ambient pressure, the leakage will be infinite. In the opposite case it will be zero, if the chamber pressure is the same as the reservoir pressure as shown in equation (7.13) and (7.14). This has to be considered in the implementation of this calculation.

\[
\lim_{p_{Ci} \to p_{amb}} \hat{A}_{Li} = \lim_{p_{Ci} \to p_{amb}} \frac{\sqrt{p_{Ci} - p_{CR}}}{\sqrt{p_{amb} - p_{Ci}}} \cdot A_{Vi} = \infty
\]  

(7.13)

\[
\lim_{p_{Ci} \to p_{CR}} \hat{A}_{Li} = \lim_{p_{Ci} \to p_{CR}} \frac{\sqrt{p_{Ci} - p_{CR}}}{\sqrt{p_{amb} - p_{Ci}}} \cdot A_{Vi} = 0
\]  

(7.14)

Beside this calculations one has to keep in mind that this estimated leakage value \( \hat{A}_{Li} \) is not only the leakage area between the chamber and the environment. There exist additional leakages among the different chambers so that they will influence each other. The center chamber e.g. has no direct connection to ambient air, but will lose negative pressure via leakage areas to the outer chambers or receive it from them.

**Behavioral Information**

The next step now is to determine a value which represents the leakage or – in other words – the amount of participation to the negative pressure system. A chamber \( i \) participates to the system if the estimated leakage of that chamber \( \hat{A}_{Li} \) is smaller than a specific threshold. In this case the maximum valve opening \( A_{V\text{max}} \) is selected as threshold, but in general it can differ from that. If the leakage value is above this threshold it can be assumed that the chamber removes energy from the reservoir which will influence also the other chambers. Therefore, activity \( a_{LE_i} \) and target rating \( r_{LE_i} \) are calculated according to equation (7.15) and (7.16):

\[
r_{LE_i} = \left( \frac{\hat{A}_{Li}}{A_{V\text{max}}} \right)_0
\]  

(7.15)

\[
a_{LE_i} = \left( \frac{\hat{A}_{Li}}{A_{V\text{max}}} \right)_0 \cdot \iota_{LE_i}
\]  

(7.16)

**7.3.2 Pressure Trend Behavior**

The second behavior supporting the chamber deactivation is an element for estimating the trend of the negative pressure. The idea is to guess the loss of negative pressure within the next program steps and to calculate the remaining time until ambient pressure would be reached. This gives an idea of the present and coming support for the adhesion system by this chamber.
First of all, the adapted pressure difference $\Delta p_{PT,i}(t)$ of chamber $i$ at current time step $t$ has to be calculated according to equation (7.17). It is adapted in a way, that it considers only a loss of negative pressure. Additionally, one of the two conditions must be fulfilled: The valve area $A_{V_i}(t)$ has been increased or remains the same compared to the previous step $t-1$ or the current pressure is larger than the desired pressure $p_{des,C_i}(t)$. Both conditions share a common background: It is obvious that the chamber pressure increases in most cases if the valve area is smaller than before. This situation is nothing to worry about. More interesting is the case if the valve area is larger and the pressure is still increasing, which is an indicator for leakages. The other condition points out, that the corresponding chamber controller is not able to achieve the desired chamber pressure.

$$\Delta p_{PT,i}(t) = \begin{cases} p_{C_i}(t) - p_{C_i}(t-1), & \text{if } p_{C_i}(t) > p_{C_i}(t-1) \land (A_{V_i}(t) \geq A_{V_i}(t-1) \lor p_{C_i}(t) \geq p_{des,C_i}(t)) \\ 0, & \text{else} \end{cases}$$ (7.17)

Based on the adapted pressure difference, the upcoming pressure difference $\hat{\Delta} p_{PT,i}(t+1)$ for the next time step(s) can be estimated. As given in equation (7.18), it uses $j$ history values and weights $w_{PT|\tau} \in \mathbb{R}^+$ with $w_{PT|\tau} > w_{PT|\tau+1}, \tau \in \{0, 1, ..., j-2\}$ and calculates the weighted average of pressure differences.

$$\hat{\Delta} p_{PT,i}(t+1) = \frac{\sum_{\tau=0}^{j-1} \Delta p_{PT,i}(t-\tau) \cdot w_{PT|\tau}}{\sum_{\tau=0}^{j-1} w_{PT|\tau}}$$ (7.18)

Via this estimated future value, the remaining time $\Delta t_{fail}^{\text{fail,C}_i}$ until the chamber will reach ambient pressure can be calculated. For this purpose, the remaining pressure difference between current chamber pressure $p_{C_i}$ and ambient pressure $p_{amb}$ is divided by the estimated pressure difference $\hat{\Delta} p_{PT,i}(t+1)$, which denotes the loss of negative pressure within one program cycle (equation (7.19)). $\Delta t$ is the cycle time of the control program.

$$\Delta t_{fail}^{\text{fail,C}_i} = \frac{p_{amb} - p_{C_i}(t)}{\hat{\Delta} p_{PT,i}(t+1)} \cdot \Delta t$$ (7.19)

It is obvious, that the estimated time – until a chamber fails – will be infinite if the estimated pressure difference is zero. This aspect has to be considered in the implementation phase of this behavior.

**Behavioral Information**

In the same manner as the leakage estimation, also the pressure trend behavior is able to trigger the chamber deactivation by its meta data. Equation (7.20) shows the calculation of the target rating $r_{PT,i}(t)$ at current time step $t$. This equation makes use of the maximum of the previous rating $r_{PT,i}(t-1)$ multiplied with an update factor $u_{PT|\tau} \in [0, 1)$ and of the quotient of current pressure trend and pressure difference threshold $\hat{\Delta} p_{PT}$. This decreasing
target rating has been chosen to allow an easier analysis of peaks and does not influence the triggering of chamber deactivation. The activity \( a_{PT_i} \) (equation (7.21)) is the ratio of current pressure trend and the maximum threshold \( \tilde{p}_{PT} \) multiplied with the behavior activation \( t_{PT_i} \).

\[
\begin{align*}
    r_{PT_i}(t) &= \max \left( \left\{ \frac{\Delta \tilde{p}_{PT_i}(t+1)}{\Delta \tilde{p}_{PT}} \right\}_0^t, r_{PT_i}(t-1) \cdot u_{PT_i} \right) \quad (7.20) \\
    a_{PT_i} &= \left\{ \frac{\Delta \tilde{p}_{PT_i}(t+1)}{\Delta \tilde{p}_{PT}} \right\}_0^t \cdot t_{PT_i} \quad (7.21)
\end{align*}
\]

### 7.3.3 Basic Chamber Deactivation Behavior

The previous sections introduced two measuring units to identify leak chambers. The next step now is to react on these situations in an optimal way. The general idea of chamber deactivation is, that if the leakage area \( A_{L_i} \) inside a chamber \( i \) becomes larger than the maximum valve area \( A_{V_{max}} \), the amount of incoming air cannot be compensated by the reservoir chamber. This is no problem in a multi-chamber-system so far, but if the suction engines cannot produce enough negative pressure, the reservoir chamber will also lose negative pressure (compare figure 7.3). In this case, also the other working chambers would be affected. Therefore, this chamber has to be cut off from the adhesion system for a certain period of time by closing the valve between working chamber and negative pressure reservoir. The basic principle of this deactivation mechanism has been introduced before as mentioned in section 2.1.3. Nevertheless, the deactivation system has been renewed to allow shorter downtimes and faster reactions on current hazards.

### Deactivation of a Chamber

The deactivation of a chamber is triggered by the two analytic behaviors, which keep an eye on the corresponding chamber: The leakage estimation behavior and the pressure trend behavior as shown in the previous sections 7.3.1 and 7.3.2. Both survey individual chamber parameters and signal their activity values through a fusion behavior to the deactivation module. The result of the fusion is a trigger value \( tr_{CD_i} \in [0, 1] \), which is the maximum of both activities \( a_{LE_i} \) and \( a_{PT_i} \) as given in equation (7.22). The single activities depend on the estimated leakage area (equation (7.16)) and on an expected pressure value as given in equation (7.21).

\[
tr_{CD_i} = \max (a_{LE_i}, a_{PT_i}) \quad (7.22)
\]

Each chamber deactivator can be in one of the three states depicted in figure 7.5. If it is inhibited or not stimulated, it remains in the inactive state, which means that it will not deactivate the corresponding chamber. This is pointed out by the two transitions to the inactive state via \( \iota_{CD_i} < 1 \). Depending on the state, a value \( s_{CD_i} \in \{-1, 0, 1\} \) is set, which is used for setting the activity and target rating of the deactivation as shown later.
Figure 7.5: Different states of the deactivation behavior: If the deactivation is active (red), the state value $s_{CD_i}$ is set to one, which will inhibit the chamber controller.

**Inactive** This is the default state of the deactivator in which the chamber remains active. It changes to active if the trigger $t_{CD_i}$ is one.

**Active** If the deactivation behavior is in the active state, the chamber will be inhibited. In this case it signals the supervisor behavior (section 7.3.4) its leakage state, which is used as a priority value for chamber testing. After a period of time, it sends a signal to start the chamber test which sets a timer $t_{CD_i}$.

**Testing** While in testing phase, the chamber is reintegrated into the adhesion system and the timer $t_{CD_i}$ is decremented by one every cycle. The testing is cancelled immediately if the estimated leakage $\hat{A}_{L_i}$ is larger than a threshold $\hat{A}_{cancel}$ which is larger than the maximum valve area ($\hat{A}_{cancel} > A_{V_{max}}$). If the timer $t_{CD_i}$ reaches zero, the chamber is either reintegrated in the case of a lower leakage area ($\hat{A}_{L_i} < A_{V_{max}}$) or the deactivator becomes active again ($\hat{A}_{L_i} \geq A_{V_{max}}$).

One additional feature has to be mentioned: As shown in figure 7.5, the deactivation behavior can switch its state via two external conditions `force active chamber` and `start chamber test`. This becomes necessary because of the distributed architecture. Here, it might e.g. happen that too many deactivation behaviors decide to shut off their chamber at the same time. In this case, the supervising behavior is able to intervene and to force the immediate reactivation of certain chambers, as examined later in section 7.3.4. There, also the second external condition to start a chamber test will be explained.

**Fast Reintegration of a Chamber without Testing**

Beside a fast reaction on dangerous situations it is also important to recover from that influence as fast as possible to reduce system stress and enhance robot safety. In general, there is a trade-off between short waiting periods (earlier reintegration, but larger stress)
and long periods (less stress, but longer time of inactivity) between the chamber tests. Requirement 14 claims a *minimized influence of the reintegration process* to keep the desired adhesion force of the system even in these critical situations. Since the leakage estimation fails, if a chamber control behavior is inhibited, each chamber has to be tested. Either it can become active again or it has to remain deactivated as it will be handled in upcoming section 7.3.4. Nevertheless, it is possible to identify inactive chambers, which can be reintegrated into the adhesion system without testing. The idea is to detect those inactive chambers which are evacuated by their active neighbors via leakages of the sealings in between the chambers. This happens, if the leakage area between chamber and ambient pressure is smaller than the leakage area to the active neighbor chambers.

![Figure 7.6: Intermediate states of a reintegration of chambers (screenshots from GUI): If chambers are influenced by a crack (a) they can only be tested periodically. Depending on basic leakages at the inner sealings a fast reintegration is possible which skips the testing phase (b,c).](image)

Figure 7.6 shows the leakage and chamber setup, if the adhesion system is influenced by a crack. In general, each inactive chamber (marked red) is tested periodically which is triggered by the chamber deactivation control behavior located in the all-embracing chamber deactivation group as depicted in figure 7.4. Depending on the leakage area caused by the crack, it is possible, that one or more chambers are shut down (red) as shown for chambers $C_3$ and $C_4$ in (figure 7.6a). These chambers will be tested periodically if they can be reintegrated into the system. Now, the robot drives away from the crack so it affects only bottom chamber $C_4$ (figure 7.6b). Normally, chamber $C_3$ now has to wait to be tested, which might take a couple of seconds, if e.g. $C_4$ is tested first. The complete procedure of testing and standard reintegration will be shown in section 7.3.4 since it is triggered from outside. But, if the pressure value of the inactive chamber decreases although its valve is still closed, the outer sealings of that chamber might be more or less leak-tight. This situation occurs, if the neighbor chamber $C_2$ and $C_7$ are able to evacuate $C_3$ via the inner sealings and their leakage areas. Finally, the robot passes the crack and also chamber $C_4$ is no longer affected, but still inactive (figure 7.6c). Here it becomes obvious, that there is a high chance that this chamber will be evacuated by its neighbors, since there exist three active neighbor chambers whose inner sealing length to $C_4$ is larger than the outer sealing of that chamber.
Algorithm 7.1: Process of a fast reintegration of an inhibited chamber without explicit testing the leakage.

1. // step 1: determine pressure values for comparison if behavior becomes active
2. if state $s_{CD_i}$ switched to active ($\{-1, 0\} \to 1$) then
3. wait until valve is closed ($A_{V_i}^{act} = 0$);
4. get pressure values of next $n$ steps;
5. calculate average $P_{C_i}^{avg}$ of last $n$ pressure values;
6. end if
7.
8. // step 2: check for reintegration while chamber is inhibited
9. while state active ($s_{CD_i} = 1$) do
10. // compare current pressure and average value
11. if $P_{C_i}^{act} < P_{C_i}^{avg} - \Delta p_{integ}^{C_i}$ then
12. // reintegrate chamber by setting deactivation inactive
13. $s_{CD_i} \leftarrow 0$;
14. end if
15. end while

The process of this reintegration procedure is given in algorithm 7.1 and starts, if the deactivation state $s_{CD_i}$ switched to active (inactive chamber). The process is stopped at any step after the waiting for the valve closure, if the deactivation behavior changes to testing phase or becomes inactive. The average value $P_{C_i}^{avg}$ is determined to compare further pressure values (in the current setup $n = 10$ steps are used). From now on, the procedure compares the current pressure value $P_{C_i}^{act}$ with the average minus a threshold value, which is currently set to $\Delta p_{integ}^{C_i} = 1000$ Pa. By this mechanism, the negative influence of the reintegration process on the adhesion system is minimized as well as the time of chamber inactivity.

Behavioral Information

The meta data of this behavior are shown in equations (7.23) and (7.24). The target rating $r_{CD_i}$ depends on the state $s_{CD_i}$ and the trigger value $tr_{CD_i}$. The behavior is unhappy, if it is active $^1$ or if the trigger value is high.

$$r_{CD_i} = \langle (s_{CD_i})_{i}^1 + tr_{CD_i} \rangle_{i}^1$$ (7.23)

$$a_{CD_i} = t_{CD_i} \cdot \langle s_{CD_i} \rangle_{i}^0$$ (7.24)

These meta data are also given to the output of the single deactivation group in which the chamber deactivation, chamber leakage estimation and chamber pressure trend behaviors are located (compare figure 7.4).

7.3.4 Chamber Deactivation Control

So far, there exist seven chamber deactivation groups. Each of them consists of a basic deactivation module with its supporting behaviors and works without any direct interaction with the other six deactivation groups. This aspect leads to two problems: At

$^1$If the valve is closed, neither the leakage nor the pressure trend can be estimated
first, too many chambers could be deactivated at the same time. Furthermore, tests for chamber reintegration could be executed in parallel, which would endanger robot safety. Therefore, a global behavior is needed which surveys the individual deactivation elements to maintain a valid state. This is done by the chamber deactivator control behavior. It is responsible for the two remaining transition conditions force active chamber and start chamber test in figure 7.5.

Guaranteed Minimum Number of Active Chambers

It is obvious, that the robot needs a certain amount of active chambers for adhesion. By default, a minimum of four chambers are declared to stay active while up to three chambers may be deactivated. To avoid a robot drop-off, the remaining chambers have to create an adhesion force of about 1500 N, which corresponds to an average negative pressure of about 96,000 Pa. Nevertheless, it has to be considered, that also the positions of the inactive chambers have a strong influence on robot adhesion, since especially the upper chambers are important for balancing the robot tilt on vertical walls. The general mechanism to guarantee this minimum makes use of the behavior-based network. Each individual deactivation behavior is inhibited with value $i_{CD_i}$ by the control behavior, as it is calculated in equation (7.25). If the number of inactive chambers – which is the sum of active ($s_{CD_i} = 1$) or testing ($s_{CD_i} = -1$) chamber deactivators – has reached the maximum of three, each inactive deactivation behavior is inhibited to remain inactive.

$$i_{CD_i} = \begin{cases} 1.0, & \text{if } \sum_{j=1}^{7} |s_{CD_j}| \geq 3 \land s_{CD_i} = 0 \\ 0.0, & \text{else} \end{cases}$$

(7.25)

This inhibition prevents other deactivation behaviors to become active, if the maximum number of inactive chambers has been reached. But, one problem still remains: If e.g. two chambers are already inactive and two additional deactivation behaviors become active at the same time, it would be possible to have only three active chambers left. Therefore, an additional check is needed to identify this situation and to force the reactivation of the chambers with lowest leakage values. This corresponds to the transition force active chamber in figure 7.5. Algorithm 7.2 shows the procedure: While too many chambers – in the current setup three of them – are inactive or in testing phase, those deactivation behaviors will be set inactive again, which just became active ($s_{CD_i}(t-1) = 0 \land s_{CD_i}(t) = 1$) and either have the lowest leakage value or – in case of a tie – whose corresponding chamber is located higher at the building. The last case has been introduced to privilege higher chambers, which contribute more to robot adhesion in terms of a better tilt balancing.

\footnote{Experiments with the prototype have shown that – in the worst case – two chambers are still sufficient for the adhesion (but not for the motion) of the robot. But, they have to be at the upper side of the robot so that it will not tilt (in normal setup this should be chambers $C_2$ and $C_6$).

\footnote{It has to be kept in mind that an active or testing deactivation behavior corresponds to an inactive chamber. So only three deactivation behaviors are allowed to be active.}
Algorithm 7.2: Force reactivation of chambers if too many deactivation behaviors became active.

1. // repeat while more than three chambers are inhibited
2. \textbf{while } \sum_{j=0}^{d} |s_{CD_j}| > 3 \textbf{ do}
3. 
4. // step 1: initialize values
5. A_{\text{min}} \leftarrow 1000000.0; // init minimum leakage
6. i_{\text{min}} \leftarrow -1; // init number of deactivator with minimum leakage
7. h_{\text{min}} \leftarrow -1.0; // init height position of minimum chamber
8. 
9. // step 2: check for new inactive chamber with lowest leakage or at heighest position
10. \textbf{for all} chambers \textbf{ do}
11. \quad \textbf{if} state s_{CD_i} switched from inactive to active (0 \rightarrow 1) \textbf{ and }
12. \quad \quad (\hat{A}_{L_i} < A_{\text{min}} \textbf{ or } (\hat{A}_{L_i} = A_{\text{min}} \textbf{ and } h(C_i) > h_{\text{min}})) \textbf{ then}
13. \quad \quad \quad // set values
14. \quad \quad \quad A_{\text{min}} \leftarrow \hat{A}_{L_i}; // update minimum leakage value
15. \quad \quad \quad i_{\text{min}} \leftarrow i; // set number of corresponding deactivator
16. \quad \quad \quad h_{\text{min}} \leftarrow h(C_i); // set height position of corresponding chamber
17. \quad \textbf{end if}
18. \textbf{end for}
19. 
20. // step 3: update state of chamber deactivator with lowest leakage
21. s_{CD_{\text{min}}} \leftarrow 0;
22. \textbf{end while}

Reintegration of Deactivated Chambers by Testing

If chambers can be deactivated they also have to be reintegrated into the negative pressure system, if the leakage value is on a normal level again. As shown before, there exists the possibility to integrate a chamber without testing it. But, this mechanism is not sufficient as given in the examples in figure 7.7. Depending on the surface structure and basic leakages between the chambers it is possible, that the leakage area between a deactivated, but integrable chamber and its active neighbors, is too low compared to the leakage area to the ambient pressure. This is obvious e.g. if there exists only one active neighbor (figure 7.7a) or if the chamber is completely isolated from the negative pressure system and has no active neighbors (figure 7.7b) – but can also occur in other situations.

![Figure 7.7](image-url)
The remaining question is now, why a testing phase is needed although a mechanism exists to estimate the leakage of each chamber. The problem lies within the leakage estimation procedure itself: In the moment of a closed valve \((A_V = 0)\) equation \((7.12)\) will result \(\hat{A}_L = 0\) which means full leak tightness. Therefore, the valve of a chamber has to be open to be able to estimate the leakage, which makes the third state (testing) of the deactivation behavior necessary. The deactivation behavior has to be switched from active \((s_{CD_i} = 1)\) to testing \((s_{CD_i} = -1)\) via the transition start chamber test as shown in figure 7.5. In a similar setup [Wettach2004] pointed out that two strategies are possible: The defensive approach is to wait longer periods of time between testing of different chambers to reduce the stress of the adhesion system. Otherwise, the chambers can be tested very fast or even overlapping to reduce the amount of unused, but tight chambers. In the developed system, a combination of aspects from both – offensive and defensive – strategies is used.

Of course, a parallel testing of chambers should be avoided to limit the stress for the remaining adhesion system, which follows the defensive approach. If a chamber deactivation behavior becomes active a timer \(t_{CD_i}\) is set to wait for reintegration of that chamber. If two or more behaviors are active, the order of testing is given by a priority value \(pr_{CD_i}\). As shown in equation \((7.26)\), this value is set initially to the estimated leakage value and is updated every cycle by an update factor \(u_{CD_{ipr}} \in (0, 1)\). This mechanism ensures that chambers with a small leakage (small priority value) are tested earlier because the chance is higher that they are air-proof again.

\[
pr_{CD_i}(0) = \hat{A}_L,
\]
\[
pr_{CD_i}(j + 1) = pr_{CD_i}(j) \cdot u_{CD_{ipr}}
\]

At next, the chambers are not simply reactivated. If the waiting time \(\Delta t_{\text{wait}}\) is over, the next chamber in the priority queue will be tested as shown in figure 7.8. Thus, the chamber will temporarily be reintegrated into the negative pressure system which is done by a state change of the corresponding deactivation behavior from active \((s_{CD_i} = 1)\) to testing \((s_{CD_i} = -1)\). Now the chamber valve opens and the current leakage value can be estimated. This is done by the leakage estimation behavior, which provides its current estimation to the chamber deactivation behavior. As given in figure 7.5, the chamber can either generate the desired negative pressure again or the leakages are still too high. If the leakage area is above a certain threshold \((\hat{A}_L > \hat{A}_{cancel})\) during the testing period \(\Delta t_{\text{test}}\) the reactivation will be aborted directly and the deactivation behavior returns to active state. Otherwise the chamber values will be analyzed until the testing period is over \((t_{CD_i} = 0)\). If no peculiar values have been measured the chamber will be marked as active again, the deactivation behavior becomes inactive \((s_{CD_i} = 0)\).

It is obvious that the system needs some time between two testing steps to allow the working chambers and the reservoir to recover from a failed test. This time \(\Delta t_{\text{CD}}\) has to be chosen carefully depending on the chosen strategy. To fasten up the reintegration process, the waiting time \(\Delta t_{\text{CD}}\) is skipped if a chamber has been reactivated successfully. In this case the next chamber is tested immediately after the previous one, which follows the offensive strategy.
Behavior-Based Adhesion Control

Figure 7.8: Exemplary process of deactivation and reintegration of chambers.

Figure 7.8 gives an example of the deactivation and reintegration process of two chambers: At the very beginning, both chambers \( i \) and \( j \) are active and the estimated leakage is below the threshold, which is given by the maximum valve areas. At timestep (A) the leakage value \( \hat{A}_{Li} \) exceeds the threshold and the chamber is deactivated, the estimated leakage goes down to zero (because of the closed valve). Now the waiting period \( \Delta t_{\text{wait}}^{CD} \) starts, but is interrupted by the failed second chamber at (B), which has a lower priority value. After the waiting time, the chamber with the lower priority value is tested again (here \( pr_{CD_i} \) at timestep (C)). During the testing period, the estimated leakage again exceeds the threshold, the chamber remains inactive (D) and the priority value is set new. At next chamber \( i \) will be tested (E) which also fails because of high leakage (F). At (G) the priority value of chamber \( j \) is smaller again and the testing period \( \Delta t_{\text{test}}^{CD} \) of the chamber starts again. During that period the leakage value remains small, the chamber can be activated. Thus, the second chamber \( i \) is tested directly after the reintegration at (H). In this case, this leakage value also does not exceed the threshold, both chambers are active again (I). Based on these multiple measures – testing cycle, cancelation of testing phase, and direct testing of next chamber – requirement 14 can be fulfilled regarding a minimum of system stress, as it is shown in the experimental results of next section 7.3.5.

Behavioral Information

This control behavior inside of the group is not only necessary for arbitration of the individual deactivation behaviors. It is also important to deliver information about the entire state of the deactivation system. Therefore, its meta data outputs are provided as group outputs.

Regarding the activity information, this behavior differs from the others. As given in equation (7.28), the activity calculation is a stair function which increases with the amount of deactivated chambers (deactivation state \( s_{CD_i} \) is either active or testing) and it becomes
fully active if three deactivation modules are active. Of course, also the chamber deactivation group is declared fully active. The group’s influence on the other behaviors like the chamber controllers reaches its maximum. Nevertheless, also the influence of the control behavior itself is at maximum while it inhibits the remaining four chamber deactivation behaviors as given in previous equation (7.25). The target rating $r_{CD}$ depends on the maximum target rating of all still inactive chamber deactivation behaviors, as shown in equation (7.27). The idea is to analyze the highest unhandled danger the adhesion system is currently confronted with. For example, this could be the highest estimated leakage value of all chambers, whose rating is one of the trigger values as shown in section 7.3.3.

$$r_{CD} = \max_{j \in \{1,2,...,7\}} \left( r_{CD_j} \cdot \left(1 - |s_{CD_j}|\right) \right)$$  \hspace{1cm} (7.27)

$$a_{CD} = \left( \frac{\sum_{j=1}^{7} |s_{CD_j}|}{3} \right)^{1/3} \cdot t_{CD}$$  \hspace{1cm} (7.28)

### 7.3.5 Experimental Results and Benefit for Safety

The benefits of the described measures like fast chamber reintegration, cancelation of testing phase and of the deactivation principle in general, are described in the following simulated experiments. At first, the general advantage of chamber deactivation is shown to proof the importance of this concept. Afterwards, some additional experiments will highlight some aspects in detail.

**Chamber Deactivation**

The general benefit of the chamber deactivation process is depicted in figure 7.9. The simulated robot drives in both experiments forward until chambers $C_1$ and $C_2$ reach a deep crack. On the left side, the system behavior without chamber deactivation is shown: At (A) the two affected chamber ($C_1$: red, $C_2$: green) lose their negative pressure. To keep the desired total downforce, also the remaining chambers have to react. Therefore, the valve of chamber $C_3$ (blue) opens shortly afterwards as shown exemplarily at (B). The problem now is that all valves are opened and the pressure inside the vacuum reservoir (black) increases as shown at (C). The plot at the bottom illustrates the decreasing adhesion force $F_{Dz}^{act}$ - the robot would drop off.

In the second case with enabled chamber deactivation, again the two frontal chambers lose negative pressure (D). In contrast to the previous case, in which the controller was not able to adjust the desired chamber pressures, these two chambers are shut down, their valves are completely closed (E). The adhesion force $F_{Dz}^{act}$ shortly drops to 1524 N, but is able to recover. Now, the inactive chambers $C_1$ and $C_2$ are tested at timesteps (F) and (G) as shown before: The valve opens and depending on the estimated leakage area the chamber is reintegrated into the system or remains inactive. This experiment shows, that a controlled shutdown of chambers is important for robot safety to maintain the desired adhesion force.
Figure 7.9: Comparison of a critical situation without (left) and with chamber deactivation (right) in which two chambers reach a deep crack at about $t = 1.1$ s.

Cancelation of Chamber Testing

In a second test series, the abort of the chamber testing phase (compare figure 7.5) of the deactivation behavior is examined in the simulation environment. Figure 7.10 shows current valve areas $A_{i}^{\text{act}}$, pressure values $p_{C_{i}}^{\text{act}}$ and estimated leakage $\hat{A}_{L_{i}}^{\text{act}}$ of three working chambers and of the vacuum reservoir. In this experiment, the simulated robot is located on rough terrain with its frontal chamber positioned on a deep crack. This situation leads to the deactivation of that specific chamber. At the very beginning of the shown plot (before time step (B)) a cyclic activation of chamber $C_{1}$ (blue graphs) can be noticed resulting from the periodical tests as given at (A). Here, the valve is opened to the maximum and the system determines the estimated leakage value. During this phase not only the first chamber is affected by the crack, but also the complete negative pressure system. The pressure reservoir (black graph) raises about 2 000 Pa during the testing phases. Furthermore, it has nearly the same negative pressure as chamber $C_{3}$ (red) so there are no further reserves remaining. Also the valve of chamber $C_{2}$ (green) remains opened completely during the whole phase, which shows that the pressure controller is at its limits. Even chamber $C_{3}$, which has no direct connection to the deactivated chamber $C_{1}$, follows the same periodic rhythm, since it has to fully open its valve (red graph) during the testing phase to maintain the desired negative pressure of about 93 000 Pa. Unfortunately, the reservoir pressure rises up to 94 000 Pa which endangers the complete system by leading to a downforce value of about 1333 N. This can lead to a drop-off of the robot, if e.g. the friction is too low. The described problem increases if more inactive chambers exists, which are indeed tested separately, but do not contribute to robot adhesion anyway.
7.3. Chamber Deactivation

Figure 7.10: Experiment with different chamber reactivation strategies: At the beginning, the test phase is not aborted (A). Afterwards (B), a test cycle is cancelled if the leakage is too high (C). After some failed tests the chamber deactivation behavior is switched from “brute-force”-testing to the enhanced cancelation via a leakage threshold $A_L^{\text{cancel}}$ as shown in figure 7.10 at (B). From now on each testing phase is aborted, if the estimated leakage is above this threshold. At (C) such a test is shown: The valve of chamber $C_1$ opens a bit, the leakage is determined (blue peak below) and the valve is closed immediately. This procedure works that fast, that the valve opens less than 10% of its maximum area until it is closed again. During this short phase, the remaining negative pressure system is nearly unaffected as it can be seen in the decreasing pressure values and valve areas of the other chambers and of the reservoir. Here, the adhesion force does not drop below 1782 N which is 450 N higher than before. This reintegration test is performed several times as given by the peaks at the estimated leakage area, before the robot is driven backwards until the frontal chamber is no longer affected by the crack. Therefore, the chamber pressure of $p_{C_1}^{\text{act}}$ decreases at (D) although the valve is still closed. At (E) the chamber is tested again with an open valve and reintegrated after a short period of time. After (F) all chambers are relaxing and adjusted to the desired chamber pressure of 93 000 Pa.
Fast Reintegration of Chambers

Figure 7.11 compares the chamber reactivation process once only with the cyclic process and once additionally with fast reintegration without testing. At the beginning of this experiment, the simulated robot is located on a crack, chambers $C_1$, $C_2$ and $C_6$ are deactivated and tested periodically as shown for one chamber if $s_{CD_6} = -1$. After some seconds the robot is driven backwards and chamber $C_2$ is no longer influenced by the crack, but still inhibited ($s_{CD_2} = 1$). In phase (A) the deactivation state with the fast integration (blue line) directly changes to inactive ($s_{CD_2} = 0$), which activates the chamber. The testing phase of the standard reactivation process has to wait until $\Delta t_{CD}^{\text{test}}$ is over, then the next chamber in the queue is checked. In this case, chamber $C_2$ is tested and successfully reintegrated – although it was fortune that none of the other chambers was tested. After each successful reactivation the next chamber is tested without a waiting phase. Here, chamber $C_1$ is considered as it can be seen at the short peak of $s_{CD_1}$ (red line) down to $-1$ at about $t = 5.5$.

![Graph showing chamber reactivation process](image)

**Figure 7.11:** Experiment showing the benefit of the fast reintegration method: Since the chambers using the reintegration (blue) become active earlier compared to the ones with classic testing cycles (red) the adhesion system is less stressed, the minimum downforce is higher.

Now, the robot is driven backward again until only chamber $C_1$ is located on the crack. Whereas the standard reactivation process has to wait $\Delta t_{CD}^{\text{wait}}$ for the next testing phase to start, chamber $C_6$ is reintegrated earlier as given by the state change (blue line) at (B). About three seconds later, the testing phase of the standard process begins at (C).
and state $s_{CDk}$ switches from 1 (chamber inhibited) to -1 (chamber testing). After the testing time $\Delta t^{test}_{CD}$ also this chamber is reintegrated. Again, the next chamber – in this case the remaining chamber $C_1$ – is tested immediately as given by the red downward peak of $s_{CD1}$. Now, this chamber is tested periodically in both cases until the robot is driven backward again. At $t = 16$ the robot completely left the crack and chamber $C_1$ is integrated immediately (D). Again, this chamber has to wait for its next testing period in the standard process.

As a result it can be stated, that the fast reintegration process reduces the time of chamber reactivation by half of the sum of waiting period $\Delta t^{wait}_{CD}$ and testing time $\Delta t^{test}_{CD}$ in average. In the given example in figure 7.11, the minimum downforce was about 200 N larger compared to the standard case. Additionally, the downforce could increase earlier. Nevertheless, it is difficult to retry these experiments with different system parameters under exactly the same conditions so these results are not repeatable.

### 7.4 Adhesion Force Control

The balancing of downforces is a fundamental demand of robot safety to avoid robot tilt and to reduce the wear of the wheels, as it has been postulated in requirement 12 (see section 5.2.2). But, the force balancing is only one part. It is obvious, that also the downforce itself has to be closed-loop controlled to avoid a robot drop-off. Therefore, two individual controllers are needed: One for the downforce value and one for its working point. Both controllers are implemented as behaviors and integrated into a behavior group, which now tries to reach a desired downforce value at the desired point and acts like a single behavior. Figure 7.12 gives an idea of these components and their relationship.

It has to be mentioned that the higher elements are related to the drive system (D) and make e.g. use of force values $F_{W1,z}$ of the wheels whereas the chamber pressure calculation and lower behaviors are associated to the negative pressure system (N) and work with chamber pressures $p_{C1}$ and valve areas $A_{V1}$.

**Figure 7.12:** Components for adhesion force control in a more detailed view compared to figure 7.1.
As input, the enclosed behaviors receive the desired force and position values $F_{D|z}$ and $\vec{P}_{F_D}$ for the drive system either from the user or from higher control elements. For closed-loop control, also current downforce $F_{act}^{D|z}$ and point of force $\vec{P}_{F_D}^{act}$ are needed, as provided by the force point calculation software module. Both values are determined according to equation (7.29) using wheel downforces $F_{act}^{W_j|z}$ and wheel positions $\vec{P}_{W_j|z}$.

$$F_{D|z}^{act} = \sum_{j=1}^{3} F_{W_j|z}^{act}$$

$$\vec{P}_{F_D}^{act} = \sum_{j=1}^{3} \vec{P}_{W_j} \cdot F_{W_j|z}^{act}$$

(7.29)

Both internal force control behaviors generate desired values $F_{N|z}$ and $\vec{P}_{F_N}$ for the negative pressure system since they use the chamber controllers as actuators. Unfortunately, these values have to be transformed to chamber pressures by the chamber pressure calculation module because they cannot be used directly. This behavior provides desired chamber pressures $p_{C_1...8}$ for each of the eight chamber control behaviors.

### 7.4.1 Force Value Control

The first behavior, which will be examined here, is responsible for the force value. In general, it is implemented as a PI-controller. The derivative portion has been left away because this leads to an unwanted oscillation of the controller because of the cascaded structure of force and pressure control. Of course, the controller has to be as fast as possible, but it should never over- or underswing as postulated in requirement 3, since this could let the robot fall down if the adhesion force is too low.

The force value controller uses the current difference value $\Delta F_{D|z} = F_{D|z} - F_{act}^{D|z}$ and calculates a desired downforce value of the negative pressure system, as given in equation (7.30). Here, $K_{FVC|p}$ and $T_{FVC|i}$ denote the amplification gain and reset time of the controller, $\Delta t$ is again the sampling time.

$$F_{N|z}(t) = K_{FVC|p} \cdot \Delta F_{D|z}(t) + K_{FVC|i} \cdot \frac{\Delta t}{T_{FVC|i}} \sum_{\tau=1}^{t} \frac{\Delta F_{D|z}(\tau) - \Delta F_{D|z}(\tau-1)}{2}$$

(7.30)

Again, the integral part is limited to a certain threshold $\dot{I}_{FVC}$ to avoid the increasing of this portion, if the negative pressure system is not able to reach the desired downforce value.

### Behavioral Information

The behavioral data of this module depend on the differences of desired $F_{D|z}$ and current downforce $F_{D|z}^{act}$. The target rating $r_{FVC}$ in equation (7.31) uses two thresholds $\dot{F}_{FVC}^{low}$ and $\dot{F}_{FVC}^{high}$ with $\dot{F}_{FVC}^{low} \leq \dot{F}_{FVC}^{high}$ to react different on too high or too low forces, respectively. Its activity is calculated as shown in equation (7.32) based on the integral portion and its limit $\dot{I}_{FVC}$.
7.4. Adhesion Force Control

\[
F_{V C} = \left\{ \max \left( \frac{F_{act} - F_D}{\Delta F_{V C}^{\text{low}}}, \frac{F_D - F_{act}}{\Delta F_{V C}^{\text{high}}} \right) \right\}_0
\]

\[
a_{F V C} = \iota_{F V C} \cdot \frac{\left| \sum_{\tau=1}^{t} (\Delta F_D(\tau) - \Delta F_D(\tau-1)) \right|}{2 \cdot I_{F V C}}
\] (7.32)

The calculation of the activity value \(a_{F V C}\) does not really fit the guidelines of the iB2C architecture because it should denote the influence of the behavior inside of the network. In this case this does not lead to conflicts because \(a_{F V C}\) is not used outside of the behavior group. Of course, the influence of the force controller should be at maximum, if it is stimulated, but this is performed by the overall force control behavior as shown later. In the existing implementation the activity shows whether the closed-loop controller is at its limits (fully active in balancing the control difference out) or not.

7.4.2 Force Point Control

The second control behavior is responsible for the position of the point of downforce – in the plane described by the contact points of the three wheels – and tries to balance it out. The goal is to distribute the adhesion force uniformly on all three wheels to reduce their wear and to avoid robot tilt. By setting the desired downforce at each wheel to an equal value \(F_W = F_{W_i}\) for wheels \(i \in \{1, 2, 3\}\) and inserting this into equation (7.29), the desired point of downforce (equation (7.33)) with wheel positions \(\vec{P}_{W_i}\) as shown in figure 2.4 can be calculated. Again, \(d_W\) is the distance between wheel and robot center.

\[
\vec{P}_{FD} = \frac{\sum_{j=1}^{3} \vec{P}_{W_j} \cdot F_{W_i}}{\sum_{j=1}^{3} F_{W_i}} = \frac{F_{W_i}}{3} \cdot \sum_{j=1}^{3} \vec{P}_{W_j}
\] (7.33)

\[
= \frac{1}{3} \cdot \left( \begin{array}{c} d_W \\ 0 \end{array} \right) + \frac{1}{\sqrt{3}} \cdot \frac{d_W}{2} \cdot \left( \begin{array}{c} \sqrt{3} \\ 0 \end{array} \right) + \frac{1}{\sqrt{3}} \cdot \frac{d_W}{2} \cdot \left( \begin{array}{c} 0 \\ \sqrt{3} \end{array} \right) = \left( \begin{array}{c} 0 \\ 0 \end{array} \right)
\]

As a result, the forces are distributed equally on the three wheels if the point of downforce lies inside of the robot center. This should be the desired position for the PI-controller in case of this specific drive setup, although other position values might also be used. The controller itself, as shown in equation (7.34), is very similar to the previous one (equation (7.30)), but works on position difference \(\Delta \vec{P}_{FD} = \vec{P}_{FD} - \vec{P}_{FD}^{\text{act}}\) with associated amplification gain \(K_{FPC_{\text{p}}}\) and reset time \(T_{FPC_{\text{i}}}\). Also the integral portion is limited again to a threshold \(I_{FPC}\).

\[
\vec{P}_{FN} = K_{FPC_{\text{p}}} \cdot \Delta \vec{P}_{FD}(t)
\]

\[
+K_{FPC_{\text{i}}} \cdot \frac{\Delta t}{T_{FPC_{\text{i}}}} \cdot \sum_{\tau=1}^{t} \frac{\Delta \vec{P}_{FD}(\tau) - \Delta \vec{P}_{FD}(\tau-1)}{2}
\]

\[
\{ \text{proportional portion, integral portion} \}
\] (7.34)
Behavioral Information

The target rating $r_{FPC}$ of this behavior depends on the difference of desired and current point of downforce, as given in equation (7.35) using a point difference threshold $\Delta \vec{P}_{FPC}$. The activity, given in equation (7.36), again depends on the integral portion and its limit $I_{FPC}$. This activity value again also does not exactly fit the guidelines of the iB2C architecture, but will not be used outside of this force control group. It behaves like the activity of the force value control behavior and grows with an increasing integral portion.

$$
    r_{FPC} = \left( \sqrt{\left(x_{act}^F - x^F\right)^2 + \left(y_{act}^F - y^F\right)^2} \right) / \Delta \vec{P}_{FPC} \\
    a_{FPC} = \iota_{FPC} \cdot \frac{\sum_{\tau=1}^{t} (\Delta \vec{P}_F(\tau) - \Delta \vec{P}_F(\tau-1))}{2 \cdot I_{FPC}}
$$

(7.35)  (7.36)

7.4.3 Force Control Group

The two force control behaviors are realized as a group as it has been shown in figure 7.12. This group contains all elements for downforce control and should act to the outside like a single behavior. Therefore, it analyzes the states of the internal behaviors and provides an overall status information for the force control system. To control the internal elements it stimulates the force value and point control modules with its own activity ($s_{FPC} = s_{FVC} = a_{FC}$). An individual inhibition or stimulation of these components is neither needed nor wanted because both elements are necessary.

Behavioral Information

The target rating $r_{FC}$ of the group is the maximum of the internal ratings, whereas the activity $a_{FC}$ is just set to the internal activation $\iota_{FC}$ (equations (7.37) and (7.38)). By this means the internal elements are not active if the force control group is inhibited.

$$
    r_{FC} = \begin{cases} 
    r_{FVC} & \text{if } r_{FVC} > r_{FPC} \\
    r_{FPC} & \text{else}
    \end{cases} \\
    a_{FC} = \iota_{FC}
$$

(7.37)  (7.38)

7.4.4 Calculation of Individual Chamber Pressures

So far, the chamber control behaviors, which have been presented in section 7.2, work on their own and not in conjunction. The last step is the transfer from an overall downforce setup as provided by the force control group to individual chamber pressures. The chamber pressure calculation (CPC) behavior performs this computation. Its task is the distribution of downforces to the seven working chambers to achieve the desired downforce value at its desired point of action. The inputs of this calculation module therefore
are the desired absolute downforce value $F_{N,z}$ (shortened to $F_N$ in the following paragraphs) of the negative pressure system and the x- and y-coordinate of the force point $\vec{P}_{F_N} = (x_{F_N}, y_{F_N})^{-1}$. As a result, the seven individual downforce values $F_{C_i}$ – one for each chamber$^4$ – have to be calculated. Finally, these forces are transferred to desired pressure values $p_{C_i}$. The single calculation steps and considerations about desired and achievable chamber forces are examined within the next paragraphs in detail. For control purposes, the behavior also has to set the desired type of chamber control as presented in section 7.2.1. In the present case the flow control mode is used for all seven working chambers so this behavior sets $c_{CC} = c_{CC}^{\text{flow}}$ for all $i \in \{1,...,7\}$.

**Direct Calculation of Desired Applied Forces**

Equation (7.39) shows the relationship between desired total force $F_N$ and the applied chamber downforces $F_{C_i}$. Additionally, the center of applied downforce $\vec{P}_{C_i} = (x_{C_i}, y_{C_i})^{-1}$ of each chamber is given to calculate the total working point of the downforce based on the desired coordinates $x_{F_N}$ and $y_{F_N}$ of the point of downforce.

\[
\begin{align*}
F_N &= \sum_{j=1}^{7} F_{C_j} \\
X_{F_N} \cdot F_N &= \sum_{j=1}^{7} \left( x_{C_j} \cdot F_{C_j} \right) \\
y_{F_N} \cdot F_N &= \sum_{j=1}^{7} \left( y_{C_j} \cdot F_{C_j} \right)
\end{align*}
\]  

(7.39)

Obviously, this equation system is overdetermined. Based on some preconditions – which are related to the geometrical setup of the robot’s adhesion system – it is possible to use equation (7.39) to calculate the desired chamber forces. The idea is to distribute the desired downforce among the chambers in a special way. Figure 7.13 illustrates the following three preconditions.

1. The center chamber $C_7$ should generate more downforce than the outer chamber in average. First, because of the slightly larger suction area. Second, because of its location: Since it lies in the robot center, it has no direct connection to the ambient air like the outer chambers and therefore can generate a higher negative pressure. Furthermore, this chamber always generated adhesion forces at the robot center.

2. The working chambers $C_1$, $C_3$ and $C_5$ can be combined to an unit, which generates half of the remaining desired downforce and pulls the point of action of the total downforce to the desired position (figure 7.13b).

3. According to the previous point the same calculation can be done for the remaining three chambers $C_2$, $C_4$ and $C_6$, which have to generate the second half of the remaining desired downforce and will also pull the point of downforce (figure 7.13c).

$^4$Due to the fact that each single chamber can produce only downforce and no turning torques the notation of downforce is simplified to $F_{C_i}$ instead of $F_{C_i,z}$.
It has to be considered that the center chamber $C_7$ produces only downforce whereas the other chambers $C_1$ to $C_6$, which are combined as shown in figures 7.13b and 7.13c, may also produce robot tilt and a specific amount of downforce in addition. Based on these aspects, it is possible to determine the downforce of the center chamber $F_{C_7}$ by using an amplification factor $\kappa_{CPC|F_C} \in (1, 2]$ according to equation (7.40). By this means, the center chamber generates more than a seventh of the total downforce without producing any unwanted tilt.

$$F_{C_7} = F_N \cdot \frac{\kappa_{CPC|F_C}}{7}$$  \hspace{1cm} (7.40)

By taking the second and third preconditions into account, it is possible to calculate the force values for the remaining outer chambers. The derivation of the final equation (7.41) describing the desired chamber force values, can be found in appendix section A.4.3.

$$F_{C_5} = \frac{(y_{FN} \cdot x_{C_1} - y_{FN} \cdot x_{C_3} - y_{C_2} \cdot x_{C_1} + y_{C_2} \cdot x_{F_{N}}) \cdot F_N}{2} + x_{C_1} \cdot y_{C_3} \cdot \frac{F_{C_7}}{2}$$  \hspace{1cm} (7.41)

$$F_{C_6} = \frac{(y_{FN} \cdot x_{C_4} - y_{FN} \cdot x_{C_2} - y_{C_2} \cdot x_{C_4} + y_{C_2} \cdot x_{F_{N}}) \cdot F_N}{2} + x_{C_4} \cdot y_{C_2} \cdot \frac{F_{C_7}}{2}$$

$$F_{C_2} = \frac{x_{C_4} - x_{F_{N}}}{2} - x_{C_1} \cdot F_{C_5} + x_{C_6} \cdot F_{C_5} - x_{C_4} \cdot \frac{F_{C_7}}{2}$$

$$F_{C_3} = \frac{x_{C_1} - x_{F_{N}}}{2} - x_{C_1} \cdot F_{C_5} + x_{C_5} \cdot F_{C_5} - x_{C_1} \cdot \frac{F_{C_7}}{2}$$

$$F_{C_1} = \frac{x_{F_N} \cdot \frac{F_N}{2} - x_{C_3} \cdot F_{C_5} - x_{C_5} \cdot F_{C_5}}{x_{C_1}}$$

$$F_{C_4} = \frac{x_{F_N} \cdot \frac{F_N}{2} - x_{C_2} \cdot F_{C_5} - x_{C_6} \cdot F_{C_5}}{x_{C_4}}$$
The final desired chamber pressure values of each chamber cannot be determined depending on the chamber’s suction area $A_{C_i}$, the desired chamber downforce $F_{C_i}$ and ambient pressure $p^{amb}$ (equation (7.42)):

$$p_{C_i} = p^{amb} - \Delta p_{C_i} = p^{amb} - \frac{F_{C_i}}{A_{C_i}} \tag{7.42}$$

Afterwards, these chamber pressure values are transmitted to the chamber control behaviors as setpoint values. Now, these components, which are responsible for closed-loop control, adjust the pressure by opening and closing the valve between working chamber and pressure reservoir, as described in previous section 7.2.

**Taking Inactive Chambers and Force Limits into Account**

Unfortunately, it is not said that all chamber controllers are able to reach the desired pressure value. The calculations, which have been shown before, give only one possible solution based on the desired values and the given preconditions. In general, there exist two possible problems in combination with the closed-loop pressure controller:

1. In the first case, the chamber should produce very low negative pressure, which is equivalent to low downforces. But, although the valve is closed, this chamber might be still evacuated by the neighbor chambers based on inner sealing leakages. In this case the chamber’s downforce is higher than desired.

2. Otherwise, the outer sealing might be not leak-proof enough or the preset of desired downforce could be too high. As a result, the chamber’s downforce is lower than desired, although the valve is already fully opened and cannot generate more negative pressure.

In both cases, the presented calculation produces solutions, which cannot be executed by the closed-loop pressure controller. Therefore, it has to be checked, if one or more chambers violate their own downforce boundaries. Unfortunately, these boundaries are not fixed. Of course, there exist theoretical limits, if the sealings are totally air-proof or the leakage is very high, but these are not helpful at this point. The idea is to handle situations, in which the controller needs to open the valves more than maximum or needs to close them more than possible. The desired force limits are calculated and provided by an additional program module (determine min/max chamber pressures) as shown in the following:

- If the current valve area $A_{V_i}^{act}$ of chamber $i$ is below a certain lower threshold $A_{V_i}^{low}$, the minimal possible downforce $F_{C_i}^{min}$ tends towards the current chamber downforce. If the valve position is zero, the minimal chamber downforce is equal to the current chamber force (equation (7.43)). $F_{C_i}^{min}$ is the theoretical absolute minimum of possible downforce – zero in general.
Behavior-Based Adhesion Control

\[ F_{C_i}^{\text{min}} = \begin{cases} 
\frac{A_{V_i}^{\text{act}}}{A_{V_i}^{\text{low}}} \cdot F_{C_i}^{\text{min}} + \left(1 - \frac{A_{V_i}^{\text{act}}}{A_{V_i}^{\text{low}}}ight) \cdot F_{C_i}^{\text{act}}, & \text{if } A_{V_i}^{\text{act}} < A_{V_i}^{\text{low}} \\
F_{C_i}^{\text{min}}, & \text{else}
\end{cases} \]

(7.43)

• Equivalent to that, a maximum downforce \( F_{C_i}^{\text{max}} \) can be calculated (equation (7.44)) with an absolute downforce maximum \( F_{C_i}^{\text{max}} \). This maximum is constrained by the chamber area and the theoretical maximum of negative pressure, which can be generated inside the chamber\(^5\). \( F_{V_i}^{\text{up}} \) is the upper threshold for the valve area, \( A_{V_i}^{\text{max}} \) is the maximum area of the valve. As in the previous case, the chambers maximum downforce tends towards the current force value, if the valve is nearly closed.

\[ F_{C_i}^{\text{max}} = \begin{cases} 
\frac{A_{V_i}^{\text{act}} - A_{V_i}^{\text{up}}}{A_{V_i}^{\text{max}} - A_{V_i}^{\text{up}}} \cdot F_{C_i}^{\text{max}} + \left(1 - \frac{A_{V_i}^{\text{act}} - A_{V_i}^{\text{up}}}{A_{V_i}^{\text{max}} - A_{V_i}^{\text{up}}}ight) \cdot F_{C_i}^{\text{act}}, & \text{if } A_{V_i}^{\text{act}} > A_{V_i}^{\text{up}} \\
F_{C_i}^{\text{max}}, & \text{else}
\end{cases} \]

(7.44)

Now, the upper and lower boundaries of each chamber are known. Based on equations (7.40) and (7.41) it can be checked, whether chambers have reached their limits or not. If the desired chamber force lies beyond one of these limits, the desired force value \( F_{C_i} \) of this chamber has to be set according to equation (7.45) and marked as bounded so that it cannot be adapted anymore.

\[ F_{C_i} = \begin{cases} 
F_{C_i}^{\text{min}}, & \text{if } F_{C_i} < F_{C_i}^{\text{min}} \\
F_{C_i}^{\text{max}}, & \text{if } F_{C_i} > F_{C_i}^{\text{max}} \\
F_{C_i}, & \text{else}
\end{cases} \]

(7.45)

Here, it can be stated, that a chamber \( C_i \) is removed out of the set of free chambers \( (S_{C|f} = S_{C|f} \setminus C_i) \) and added to the pool of bounded chambers: \( S_{C|b} = S_{C|b} \cup C_i \). It should be mentioned, that a chamber is either bounded or free and that all chambers have to be within one of both sets: \( S_{C|f} \cap S_{C|b} = \emptyset \) and \( S_{C|f} \cup S_{C|b} = S_C = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7\} \).

Optimization of Force Values

The question is now, whether the system still reaches the desired downforce value and working point or not. As it can be imagined, both desired inputs will not be achieved if one or more chamber downforce values \( F_{C_i} \) are changed without any balancing update of the opposite or neighbor chambers. Therefore, an iterative optimization algorithm has to be performed, which adapts the remaining adjustable (free) chambers in their downforce value. This optimization method is based on the calculation of pressure updates mentioned in section 2.1.3, but now uses force values instead of pressure values and extends

---

\(^5\)To be precise, this maximum differs from one chamber to another if the chamber areas are different. In this case it is valid to use only one downforce maximum as only the center chamber is slightly larger than the outer chambers. In practical application, this maximum value cannot be reached by the chambers because of leakages and other disturbances.
it concerning the bounding of chambers, which is important for optimal downforce results and robot safety. The first optimization step takes the results from equation (7.45) and sets the initial forces to $F_{C_i}(0) = F_{C_i}$.

Based on the desired force values of all seven working chambers, the resulting downforce $F_N(j)$ and its working point $\vec{P}_{F_N}(j)$ in optimization step $j$ have to be calculated according to equation (7.46) and (7.47).

$$F_N(j) = \sum_{i=1}^{7} F_{C_i}(j)$$ (7.46)

$$\vec{P}_{F_N}(j) = \begin{pmatrix} x_{F_N}(j) \\ y_{F_N}(j) \end{pmatrix} = \sum_{i=1}^{7} \vec{P}_{C_i} \cdot \frac{F_{C_i}(j)}{F_N(j)}$$ (7.47)

Both values are needed to calculate the differences between these optimized values and the desired ones. An optimization method is used to minimize the differences. To achieve this, it adapts the force values of those chambers, which are not bounded to any limits. The influence of each free (unbounded) chamber to adapt the force position $\vec{P}_{F_N}(j)$ can be calculated by taking the chamber center of downforce $\vec{P}_{C_i}$, the desired center position $\vec{P}_{F_N}$, and the center $\vec{P}_{F_N}(j)$ of iteration step $j$ into account (see figure 7.14). This influence can be calculated using the distance $\Delta P_{|C_i,F_N|}$ from chamber center to desired downforce point (equation (7.49)) and the distance $\Delta P_{|C_i,F_N(j)|}$ of chamber center to optimized downforce point (equation (7.48)). In this example chamber $C_3$ has a large influence on the position of the force point (figure 7.14b). It has to reduce the negative pressure to push the force point (red) into the desired direction (green), whereas the opposite chambers have to increase their adhesion force. In contrast to that the influence of chamber $C_2$ can be neglected ($u_{F_{C_2}}(j)$ in figure 7.14a is close to zero).

$$\Delta P_{|C_i,F_N|} = \sqrt{(x_{C_i} - x_{F_N})^2 + (y_{C_i} - y_{F_N})^2}$$ (7.48)

$$\Delta P_{|C_i,F_N(j)|} = \sqrt{(x_{C_i} - x_{F_N(j)})^2 + (y_{C_i} - y_{F_N})^2}$$ (7.49)

Based on these two distances, an update value $u_{F_{C_i}}(j)$ can be calculated. This value is the difference of both distances according to equation (7.50), which is used within one optimization step to update the desired chamber force value. This update value can only be applied to unbounded chambers which still have a degree of freedom in their downforce, and is a measure for the influence of a chamber to optimize the point of downforce. Figure 7.14 shows the geometrical relationship between both force center points, the location of the chamber centers and the update value.

$$u_{F_{C_i}}(j) = \Delta P_{|C_i,F_N(j)|} - \Delta P_{|C_i,F_N|}$$ (7.50)
The individual chamber forces \( F_{C_i}(j) \) are then updated regarding equation (7.51) with a constant amplification factor \( \kappa_{PC\mid F_C} \in \mathbb{R}^+ \) and an intermediate index \( j' \). This factor determines the step size of the optimization and therefore how fast the iterative algorithm converges. If the desired chamber force value had already reached its upper or lower limit according equation (7.45), this force is bounded and cannot be changed by the update factor. Nevertheless, it is obvious, that the desired chamber force values of unbounded chambers are not the final ones.

\[
F_{C_i}(j') = \begin{cases} 
F_{C_i}(j) & \text{if } C_i \in \mathbb{S}_{C|b} \text{ (chamber is bounded)} \\
F_{C_i}(j) + \kappa_{PC\mid F_C} \cdot u_{F_{C_i}}(j) & \text{else}
\end{cases}
\]  

(7.51)

In general, the intermediate sum of downforces \( F_N(j') \) (from equation (7.46)) will still differ from the desired value \( F_N \) after this adaption. Remember, that the previous steps are used to optimize only the point of downforce. But, unfortunately, also the total desired downforce value is affected if one or more chambers reached their limits, which will raise or lower the total force. In this case the force values \( F_{C_i}(j') \) of the free chambers have to be adapted to reach the desired downforce. For this reason, the temporary amount of free (or unbounded) force \( F_{N|f}(j') \) and the bounded one \( F_{N|b}(j') \) have to be calculated according to equation (7.52).

\[
F_{N|f}(j') = \sum_{C_i \in \mathbb{S}_{C|f}} F_{C_i}(j') \quad F_{N|b}(j') = \sum_{C_i \in \mathbb{S}_{C|b}} F_{C_i}(j')
\]  

(7.52)
\[ F_N(j') = F_{N|f}(j') + F_{N|b}(j') = \sum_{i=1}^{7} F_{C_i}(j') \] (7.53)

All these sums might change during each optimization step because of additional bounded chambers or invalid intermediate values. Of course, only the desired forces of the free chambers can be changed. Vice versa the optimization finishes, if all chambers are bounded. In this case equation (7.52) results in \( F_{N|f}(j') = 0 \) and the intermediate values \( F_{C_i}(j') \) are also the final ones. If not all chambers are bounded, the force values of the remaining free chambers have to be adapted regarding equation (7.54) while \( F_N(j') \neq F_N \). Here, the chamber force is updated with the ratio of desired sum of free downforces \( (F_N - F_{N|b}(j')) \) and current free downforces \( F_{N|f}(j') \):

\[
F_{C_i}(j + 1) = \begin{cases} 
F_{C_i}(j') \cdot \frac{F_N - F_{N|b}(j')}{{F_{N|f}(j')}} & \text{if } C_i \in S_{C|f} \text{ (chamber is free)} \\
F_{C_i}(j') & \text{else}
\end{cases}
\] (7.54)

Now, the desired and the optimized total downforce are equal \( (F_N(j + 1) = F_N) \) except the case of completely bounded chambers. In this case the desired target downforce cannot be reached. It might also happen that after the force update in equation (7.54) one or more free chambers have reached their limits. Now their forces have to be set according to equation (7.45) and the current optimization step has to be repeated with the new bounded chambers. This inner optimization cycle continues, until either all chambers are bounded or no new chambers have been bounded.

If the inner optimization cycle is completed, the resulting downforce value \( F_N(j + 1) \) and point \( \vec{P}_{F_N}(j + 1) \) with the current settings must be calculated, as it is shown in equations (7.46) and (7.47). The results have to be compared to the desired values to determine, if another optimization step should be performed or if the optimization algorithm should terminate. The used optimization function \( f_{CPC}(j + 1) \) itself is shown in equation (7.55). Its parameter \( w_{f_{CPC}} \in [0, 1] \) is used as ratio between downforce and position differences to allow a weighting of both terms.

\[
f_{CPC}(j + 1) = w_{f_{CPC}} \cdot \frac{\text{downforce difference}}{F_N - F_N(j + 1)} \\
+ (1 - w_{f_{CPC}}) \cdot \sqrt{(x_{F_N} - x_{F_N}(j + 1))^2 + (y_{F_N} - y_{F_N}(j + 1))^2}
\] (7.55)

Finally, there exist several criteria depending on the system’s state for a termination of the optimization process:

**Success** The optimization (nearly) reaches the desired values: \( f_{CPC}(j + 1) \approx 0 \). Here a threshold can be used to allow a small amount of inaccuracy.
**Local minimum** The previous calculation provided better results for downforce and center point \( f_{CPC}(j + 1) > f_{CPC}(j) \). In this case the results from the previous step have to be used.

**Optimization threshold** An optimization step threshold \( \Delta \hat{f}_{CPC} \) has been under-run, so that the enhancement is very small \( f_{CPC}(j) - f_{CPC}(j + 1) < \Delta \hat{f}_{CPC} \).

**Iteration limit** The maximum number of iterations has been reached.

**(Nearly) all chambers bounded** As described above, the optimization has (nearly) no freedom to change chamber force values, if more or less no free chambers are left: \( F_{N|f}(j + 1) \approx 0 \)

The complete algorithm which performs the optimization of negative force distribution can be seen in algorithm 7.3. It follows the explanations given in the previous paragraphs and shows how the force calculation can be implemented. Finally, the optimized chamber force values \( F_C(n) \) of final step \( n \) have to be used to calculate the desired chamber pressures \( p_C \) according to equation (7.42). These values will be given to the chamber control behaviors shown in section 7.2.

### Behavioral Information

As shown before, this behavior calculates individual chamber pressure values to generate a desired downforce at a specific point of action. For the behavior-based analysis of the state of the negative pressure system it is important to know, when behaviors are unsatisfied or in which way they contribute to the whole control process. In this case the behavior cannot be activated manually because it would not make any sense without corresponding desired force values. Therefore, this behavior is triggered by force controlling behaviors, which generate the desired commands either by manual user control or via the force control group shown in previous section 7.4.3. For reasons of further processing of the calculated pressure values, the activity \( a_{CPC} \) is set to the internal activation \( \iota_{CPC} \) of this behavior as shown in equation (7.57). The target rating \( r_{CPC} \) (equation (7.56)) uses the maximum of two individual ratings\(^6\): The first one \( r_{CPC|F} \) evaluates the difference between calculated and desired downforce value whereas \( r_{CPC|P} \) rates the displacement of the point of downforce. By this means \( r_{CPC} \) expresses the theoretical ability of the negative pressure system to achieve the desired force values. If all chambers are working and the leakages are not that high, the behavior will find a suitable distribution of the individual chamber downforces. Disturbances like cracks or impossible desired values will lead to an unsatisfied behavior, if the chambers cannot reach the desired force values anymore.

\[
\begin{align*}
    r_{CPC} &= \max \left( r_{CPC|F}, r_{CPC|P} \right) \\ 
    a_{CPC} &= \iota_{CPC}
\end{align*}
\]

\(^6\)For testing purposes it is also possible to use the limited sum or the product of both ratings.
7.4. Adhesion Force Control

Algorithm 7.3: Process of optimizing chamber downforces.

1. // step 1: initialize values
2. for all $C_i \in S_C$ (all chambers) do
3.   calculate initial desired chamber force value $F_{C_i}(0)$; // equations (7.40) and (7.41)
4.   calculate minimum and maximum values $F_{C_i}^{\text{min}}$ and $F_{C_i}^{\text{max}}$; // equations (7.43) and (7.44)
5. end for
6. $S_{C|f} \leftarrow S_C$; // put all chambers into the free set
7. $S_{C|b} \leftarrow \emptyset$;
8. $j \leftarrow 0$; // optimization step
9. finished $\leftarrow$ true; // initial set to indicate end of optimization process
10. // main loop, repeated until desired values or the optimum reached
11. repeat
12.   // step 2: perform optimization step if not finished and minimum of two free chambers exist
13.   if finished $= false$ and $|S_{C|f}| > 1$ then
14.     // calculate updates of chamber downforces
15.     for all $C_i \in S_{C|f}$ (free chambers) do
16.       $F_{C_i}(j') \leftarrow F_{C_i}(j) \cdot \kappa_{CPC}|F_{C} \cdot u_{C_i}(j)$; // equation (7.51)
17.       end for
18.     $F_{N|f}(j') = \sum_{C_i \in S_{C|f}} F_{C_i}(j')$; // equation (7.52)
19.     $F_{N|b}(j') = \sum_{C_i \in S_{C|b}} F_{C_i}(j')$; // equation (7.52)
20.     for all $C_i \in S_{C|f}$ (free chambers) do
21.       $F_{C_i}(j + 1) \leftarrow F_{C_i}(j') \cdot \frac{F_{N|f}(j')}{F_{N|f}(j')}$; // equation (7.54)
22.       end for
23.   end if
24. end if
25. // step 3: check force limits of chambers
26. finished $\leftarrow$ true;
27. for all $C_i \in S_C$ (all chambers) do
28.   if $F_{C_i}(j) > F_{C_i}^{\text{max}}$ or $F_{C_i}(j) < F_{C_i}^{\text{min}}$ then
29.     $F_{C_i}(j + 1) \leftarrow F_{C_i}(j') \leftarrow \min(F_{C_i}^{\text{max}}, \max(F_{C_i}^{\text{min}}, F_{C_i}(j)))$;
30.     finished $\leftarrow$ false; // set, to execute further optimization step(s)
31.     $S_{C|f} \leftarrow S_{C|f} \setminus C_i$; // remove chamber out of free set
32.     $S_{C|b} \leftarrow S_{C|b} \cup C_i$; // add chamber to bounded set
33.   end if
34. end for
35. // step 4: check for execution end if no new chambers have been bounded
36. if finished $= true$ then
37.   calculate optimization function $f_{CPC}(j + 1)$; // equation (7.55)
38.   finished $\leftarrow$ false;
39. if $f_{CPC}(j + 1) \approx 0$ or $f_{CPC}(j) - f_{CPC}(j + 1) < \Delta f_{CPC}$ or $j > $ limit then
40.   finished $\leftarrow$ true;
41.   $n \leftarrow j + 1$; // use current values as final ones
42. end if
43. if $f_{CPC}(j + 1) > f_{CPC}(j)$ then
44.   finished $\leftarrow$ true;
45.   $n \leftarrow j + 1$; // use previous values as final ones
46. end if
47. $j \leftarrow j + 1$;
48. end if
49. until finished
The underlying ratings from equation (7.56) are calculated according to equations (7.58) and (7.59). Both make use of three additional threshold values: \( \Delta F_{\text{CPC}}^{\text{max}} \) is the distance limit, whereas \( \Delta F_{\text{CPC}}^{\text{low}} \) and \( \Delta F_{\text{CPC}}^{\text{high}} \) are force thresholds in case of too high and too low forces, respective, with \( \Delta F_{\text{CPC}}^{\text{low}} \leq \Delta F_{\text{CPC}}^{\text{high}} \). The reason for these two values is, that the penalty for too low forces in case of \( F_N > F_N(n) \) should be larger than for high forces.

\[
r_{CPC|F} = \left( \max \left( \frac{F_N - F_N(n)}{\Delta F_{\text{CPC}}^{\text{low}}}, \frac{F_N(n) - F_N}{\Delta F_{\text{CPC}}^{\text{high}}} \right) \right)^{1/0} \tag{7.58}
\]

\[
r_{CPC|P} = \left( \sqrt{(x_{F_N}(n) - x_{F_N})^2 + (y_{F_N}(n) - y_{F_N})^2} \right) / \Delta F_{\text{CPC}}^{\text{max}} \tag{7.59}
\]

### 7.4.5 Experimental Results and Benefit for Safety

The transfer from desired downforce to desired chamber pressures is essential for adhesion control. Figure 7.15 compares the presented direct calculation method with two former approaches by [Wettach2004] and [Hillenbrand2009]. In this experiment the simulated robot is driven downwards a wall, the desired downforce values of the negative pressure system are set to \( F_N(z) = 2700 \text{ N}, x_{F_N} = 0.036 \text{ m} \) and \( y_{F_N} = 0.0 \text{ m} \). The graphs show the absolute differences between desired and current values as given in equation (7.60) – the lower the better. At the beginning the robot drives on a rough patch of surface, but nearly all approaches are able to adjust the desired downforce value and point of downforce. Only the x-position in the second approach is not located around zero which is a result of a limitation of iteration cycles. This control difference can be reduced by additional iterations, which comes at the costs of an increased calculation time (compare table 7.1).

\[
\Delta x_{F_N} = |x_{F_N}^{\text{act}} - x_{F_N}|, \quad \Delta y_{F_N} = |y_{F_N}^{\text{act}} - y_{F_N}|, \quad \Delta F_{N|z} = |F_{N|z}^{\text{act}} - F_{N|z}| \tag{7.60}
\]

At about \( t = 11.5 \text{ s} \) the frontal chamber reaches a crack and is deactivated. As a result, all controllers are affected by this event as it can be seen in the increasing control differences. Nevertheless, the direct calculation is performing outstanding compared to the other approaches. Not only the displacement of the downforce point is 32\% respective 40\% lower. The drop of downforce of 390 N is neglectable compared to a loss of at least 904 N. Here, the robot is in large danger of losing too much adhesion force and to drop off. The results show, that the new direct calculation method performs significant better than the existing approaches and supports robot safety in an eminent manner.

Beside the general benefit of considering chamber limits leading to a better control performance, the presented approach is also much faster compared to existing methods as shown in table 7.1. In average a speedup of factor six compared to the approach of [Hillenbrand2009] and of factor 14.5 compared to [Wettach2004], respectively, could be achieved. This is important to guarantee the allocation of the control data within the next program cycle.
Figure 7.15: Plots of differences between desired and current force values (position and down-force) of three different approaches. One can see that the direct pressure calculation provides best results if at least one chamber is deactivated starting at $t = 11.5$ s.

Table 7.1: Comparison of the presented direct calculation method with two existing approaches which always need a couple of iterations for optimization.

<table>
<thead>
<tr>
<th>calculation method</th>
<th>[Wettach2004]</th>
<th>[Hillenbrand2009]</th>
<th>direct calculation$^a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg. computation time</td>
<td>0.127 ms</td>
<td>0.052 ms</td>
<td>0.009 ms</td>
</tr>
<tr>
<td>max. computation time</td>
<td>2.563 ms</td>
<td>0.096 ms</td>
<td>0.058 ms</td>
</tr>
<tr>
<td>avg. iterations</td>
<td>49.9</td>
<td>50.0</td>
<td>2.82</td>
</tr>
<tr>
<td>max. iterations</td>
<td>$50^b$</td>
<td>$50^b$</td>
<td>27</td>
</tr>
</tbody>
</table>

$^a$ The presented calculation method is a combination of direct calculation and additional (optional) optimization steps if the chamber limits are reached.

$^b$ This is the predefined maximum of optimization cycles for all three methods.

7.5 Adhesion Network and Behavior Interaction

So far, the algorithmic components and logical units of the adhesion control system have been presented as well as the calculations of the meta data. To complete the description of the control network, this section will focus on the collaboration of the single control elements and introduce some additional behaviors [Schmidt2011a]. Figure 7.16 depicts the complete adhesion control network in iB2C view (compare section 2.2.1) reaching from the force control group on top down to the chamber controllers and the chamber deactivation behaviors. The structure consists of four different levels: Force control, pressure control, chamber control and chamber deactivation.
Figure 7.16: Setup and interaction of the behavior network. The control data flow is given as thick gray line (for clearness the sensor data is not shown here). The structure of the internal groups are depicted in figures 7.17 and 7.18.
By default, the individual chamber pressures are controlled manually on startup to allow a check-up of the system. During operation all elements should be active for safety aspects. Those elements without a connected stimulation input like the manual pressure control or fusion behaviors are always stimulated.

The overall behavior-based adhesion network combines distributed control elements and additional components with a higher view for arbitration as in the case of the chamber deactivation network. In fact, there exists no complete model of the dynamic adhesion process. Instead of that, the knowledge is distributed among the components so that each of them is able to perform the required actions to achieve the desired goal. Each component does not necessarily know much about the elements next to it. The force controller for example does not know which chamber has been shut down. In the same manner each pressure control behavior does not know what total amount of adhesion force is demanded. The knowledge about the processes for this complex adhesion control is distributed, so that each component receives exactly the portion of knowledge, which is needed for its task execution.

### 7.5.1 Force Control

The most convenient feature of the adhesion control network is the force control. As described in section 7.4, it is used for closed-loop control of the downforces at the wheels measured by the load cells. Figure 7.17 shows the internal structure of the force control group. It consists of the two behaviors, which are responsible for closed-loop downforce and force point control, and a general force control behavior. This behavior collects the meta data of the two controllers and determines an overall target rating value which is given to the group’s output. It also stimulates the two behaviors via its own activity, whereas the whole group can be inhibited from outside (which is not used so far).

![Figure 7.17](image-url)

**Figure 7.17:** Content of the force control group: Two control behaviors and one for generating group’s meta data.

But, for debugging purposes the user must also be able to set ‘uncontrolled’ downforce values which are simply adjusted by the adhesion system without a closed loop. Since the chamber pressure calculation behavior takes desired force values as inputs, two additional elements are needed as depicted in figure 7.16: manual force control and a force fusion behavior.
Manual Force Control

At first, a new behavior is needed to allow the manual setting of a desired adhesion force. The *manual force control* behavior just forwards the desired force values to the output. Its target rating $r_{MFC}$ is calculated similar to the values of the closed-loop force control behaviors from section 7.4.1 and 7.4.2. As given in equation (7.61), it is the maximum of a force point rating depending on the current and desired coordinates $x_{FN}$ and $y_{FN}$ related to the negative pressure system and of the downforce $F_{N|z}$. Again, $\Delta P_{MFC}$, $\hat{F}_{MFC}^{\text{high}}$ and $\hat{F}_{MFC}^{\text{low}}$ are certain thresholds for position and downforce discrepancy. In other words: This behavior is satisfied, if the underlying negative pressure system can reach the desired values. If the values are too extreme or in cases of high leakages the behavior becomes unsatisfied, although it has no ability to react on this situation. The *activity* is simply set to the internal activation $\iota_{MFC}$ as shown in equation (7.62).

$$r_{MFC} = \max \left( \left\langle \left( \frac{(x_{act}^{\text{get}} - x_{FN})^2 + (y_{act}^{\text{get}} - y_{FN})^2}{\Delta P_{MFC}} \right) \right\rangle_0, \right.$$  
$$\left\langle \max \left( \frac{F_{act|z}^{\text{get}} - F_{N|z}}{\Delta F_{MFC}^{\text{high}}}, \frac{F_{act|z}^{\text{get}} - F_{act|z}^{\text{get}}}{\Delta F_{MFC}^{\text{low}}} \right) \right\rangle_0 \right)$$  

(7.61)

$$a_{MFC} = \iota_{MFC}$$  

(7.62)

This behavior can be stimulated by the user in the same way as the *force control group*. But, in contrast to that, it is inhibited by the activity of this group. By this mechanism a kind of priorization is implemented, which favors the closed-loop force controller.

Force Fusion Behavior

So far, two force controlling behaviors have been introduced which try to give desired force values to the negative pressure system. Therefore, an additional fusion behavior is needed, which merges the two inputs into one output as depicted in figure 7.16. Based on the activity values, the *force fusion* forwards the values of the more active behavior via maximum fusion. By its own activity – which is the maximum of the incoming activities – it stimulates the *chamber pressure calculation* behavior to transfer the overall force values to individual chamber pressures.

7.5.2 Chamber Control

The *chamber pressure calculation* behavior provides individual desired chamber pressures, which are given to the *chamber control* group. But, in the same way as before, additional elements are needed to set the pressure values manually e. g. for debugging purposes.

Manual Chamber Control

One of these additional behaviors is the *manual chamber control*. It forwards the inputs from the user and sets its meta data according to equations (7.63) and (7.64). The target
rating \( r_{MCC} \) is set fix to 0.5, which corresponds to the iB2C guidelines in the special case, that a behavior has no meaningful satisfaction value. As given in section 7.2.1, the chamber controllers are able to adjust the chamber pressure in two different ways (flow control and pressure control), but also to set the valve opening or to initialize the stepper motors. Therefore, this behavior forwards the desired manual pressure values \( p_{C_i} \), valve openings \( A_{V_i} \) and controller settings \( c_{CC_i} \).

\[
\begin{align*}
    r_{MCC} &= 0.5 \\
    a_{MCC} &= t_{MCC}
\end{align*}
\]  

(7.63) \hspace{1cm} (7.64)

As depicted in figure 7.16, this behavior is always stimulated\(^7\) since it is an elementary controller. But, it is inhibited by the activity of the force fusion behavior. Via this connection the manual chamber control becomes inactive if one of the higher force controllers is used.

Chamber Fusion Behavior

Again, two behaviors try to access the same resource. In this case the chamber control group should be used either by the calculation behavior or by the manual control behavior. Therefore, another fusion behavior is needed to merge the outputs of both modules. The chamber fusion behavior is a maximum fusion again forwarding the values of the behavior with the higher activity to the chamber control behaviors.

7.5.3 Chamber Controllers and Deactivation Behaviors

As described before, the chamber controllers are the basic component for robot adhesion. The chamber control group (figure 7.16) acts like a single behavior, although some characteristics are different: For instance, the group is always stimulated and cannot be inhibited by other modules since it is essential for adhesion\(^8\). The internal chamber control behaviors are stimulated via the general input interface for debugging purposes. Again, the chamber controllers must be stimulated during operation. The chamber control behavior collects the meta data \((a \text{ and } r)\) of the individual behaviors and determines overall values for activity and target rating, which represent the complete negative pressure system (see section 7.2.2).

The only chance of curbing a chamber control behavior is the inhibition via the corresponding chamber deactivation group. Figure 7.16 shows the connection of the activity output of each single chamber deactivation group to the inhibition input of the chamber control behavior. The deactivation group itself becomes active via user stimuli, so the robot operator can decide if he wants to use the emergency shutdown system or not. The deactivation control behavior (section 7.3.4) calculates group meta data and inhibits deactivation behaviors to prevent the shutdown of too many chambers.

\(^7\)Remember: A missing stimulation input connection represents the default stimulation of this behavior with the maximum stimuli of 1. Otherwise the behavior could never become active and might be useless within the network.

\(^8\)Of course, this inhibition is possible in theory due to the structure of the behavior-based network. But, this should be prohibited in practice.
7.5.4 Experimental Results

To show the mode of operation of the presented behavioral interaction, an experiment has been executed in a simulated environment [Schmidt2011a] which describes, how the different behaviors influence each other. Since the behavior-based system should work in reality as well as in the simulated environment, these experiments have been performed in simulation to check if the network works in principle.

Experimental Setup

In this experiment the simulated climbing robot drives downwards a wall facing a deep crack as depicted in figure 7.19. The right images are screenshots from the graphical user interface (GUI) showing the simulated scene including the robot, a structured wall and some decorations in the background. On the left side the chamber state in form of graphical representations is illustrated: The upper symbol describes the individual chamber pressures as colors reaching from dark green (very high negative pressure) to red (ambient pressure). Three black circles represent the downforces at the wheels by their diameter. The chamber colors of the symbol below illustrate the state of the chambers which can be active (green), inactive (red) or testing (yellow). During the test run, the robot was driven downwards (figure 7.19a) until the three frontal chambers $C_1$, $C_2$ and $C_6$ are above a crack and had to be shut off (figure 7.19b). Afterwards, the robot is driven backwards up the wall (figure 7.19c) since this deep crack cannot be overcome by the robot.
Figure 7.19: Intermediate steps of the experiment: At first, the robot is driven downwards (a). If three chambers are located on a deep crack the robot stops (b) and drives backwards again (c). During the test run, up to three chambers are shut off (deactivated).

Results of Behavior Interaction

The behavioral interaction itself is shown in figure 7.20. On the left side activity (green graphs) and activation values (red graphs) of the behaviors are given. On the right, the target rating values of each behavior (blue graph) are plotted. Since the complete network consists of 47 behavioral modules, groups and fusions, not all of them can be depicted here. For instance, the individual behaviors of chambers $C_4$ to $C_7$ are not shown since their characteristics are similar to the others. At the beginning of the described test run, the manual chamber controller (MCC) is active ($a_{MCC} = 1$) with an uniform chamber pressure for all chambers. Therefore, robot tilt is not balanced out which results in a x-position of the downforce point $P_{FD|x}^{act}$ of about 0.042 m as shown in figure 7.21 on top (red graph at $t = 0$ s).
Figure 7.20: Plot of behavior meta data activity $a$ (green), activation $i$ (red) and target ratings $r$ (blue). Corresponding downforce values are depicted in figure 7.21.

At (A), the user stimulates the manual force controller ($a_{MFC} = i_{MFC} = 1$), which inhibits the MCC ($a_{MCC} = i_{MCC} = 0$). Since now a transfer from force values to pressure values is needed, the MFC additionally stimulates the chamber pressure calculator (CPC). Both target rating values $r_{MFC}$ and $r_{CPC}$ are satisfied because the system is able to generate the desired adhesion forces. This is easy so far, because they take only the negative pressure system into account without considering wheel’s downforces or robot tilt. At (B) the force control group (FC) is stimulated ($a_{FC} = i_{FC} = 1$), which tries now to balance
wheel forces with a desired total downforce of $F_{D|z} = 2200 \text{ N}$. This group makes also use of the CPC, so this behavior remains stimulated while MFC and MCC are inhibited ($\iota_{MFC} = \iota_{MCC} = 0$). This situation correlates to figure 7.19a with balanced downforces of all three wheels, although the target ratings of the participating behaviors show that it is not easy to reach this state.

Now the robot tries to overcome the crack, but the leakages are too high. At (C) the frontal chamber $C_1$ needs to be shutdown by the corresponding chamber deactivator behavior. $cd_i$ becomes active ($a_{CD_i} = 1$), which inhibits the chamber controller ($\iota_{CC_i} = 0$). Chambers $C_2$ and $C_6$ follow the same way at (D), until the complete robot front reaches the crack (figure 7.19(b)). The red graphs of the three corresponding (and inhibited) chamber controllers go down to zero. Now the chamber deactivation reaches its maximum activity ($a_{CD_i} = 1$) because three deactivation behaviors are active. Therefore, the other deactivation modules $cd_3, cd_4, cd_5$ and $cd_7$ are inhibited to avoid a shutdown of additional chambers, which can be seen in the internal activation value $\iota_{CD_1}$ at the bottom of figure 7.20 which goes down to zero.

![Figure 7.21: The corresponding force values to figure 7.20: The x- (red) and y-position (green) of the downforce point and the resulting downforce $F_{D|z}^{\text{act}}$ (blue graph).](image)

At step (E) the adhesion system tries to reintegrate the first chamber $C_1$ into the system by testing it. For a short period of time $a_{CC_1}$ is set to one – the valve opens again – before it is inhibited again by the deactivation element because of too high leakages, since the chamber is still influenced by the crack. This testing cycle is also visualized in figure 7.19b with the yellow marked frontal chamber on the left side. The minimum downforce in this period is at about 1760 N provided by the four remaining chambers as depicted in figure 7.21. The minimum x-coordinate of the downforce point moved to $-0.082 \text{ m}$, which is still within the stability triangle so the robot does not tilt.
Now the robot is driven backward up the wall, since the crack cannot be handled by the adhesion system (figure 7.19c). At (F), chamber $C_2$ was reintegrated successfully into the system which in turn removes the inhibition of the remaining chamber deactivation behaviors ($t_{CD_3} = 1$). The frontal chamber follows at timestep (G), the force and chamber controllers can recover and the downforce point $\vec{P}_{FD}^{\text{act}}$ returns to a value around zero.

### 7.5.5 Benefit for Robot Safety

Again, a closed-loop controller itself or any other algorithmic element does not become faster, more reliable or more accurate if it is implemented as a behavior. The ideas behind behavior-based approaches are more focussed on the embedding network consisting of a couple of behaviors and their interactions, as shown in figure 7.16. Therefore, the individual chamber control behaviors are not better than classic implementations of the same controllers. The benefit regarding robot safety is based on the online analysis via additional behavioral meta data and on the way of developing and integrating new control behaviors.

![Figure 7.22: Annotated screenshot of the chamber control group visualized via mca browser, which allows a detailed view on the structure of the control network.](image)

Figure 7.22 shows a screenshot of the mca browser, which is a special tool to take a look at the internal program structure and to update certain program parameters (see section A.6.2 for more details). If iB2C behaviors are used, this analysis tool can visualize internal states of the behaviors and how they interact. The figure presents the chamber control group with its internal behaviors consisting of seven behaviors for chamber control named Chamber Control 1 to 7, as introduced in section 7.2.1, and the Chamber Pressure State behavior, which determines the state of the overall negative pressure system (section 7.2.2). This figure illustrates the stimulation connections, which are marked green coming from the group’s controller input interface. In the same way it is e.g. possible to determine causes of missing activity of a behavior, which might be caused because of a missing stimulus or because of an inhibition from outside. A trace back of the source of this inhibition can be done easily. In this presented case, Chamber Control 1 is influenced by the corresponding chamber deactivation behavior, which is located outside of this
group. The colored bars at each behavior node represent the individual states activation (yellow), activity (green) and target rating (red) reaching from 0 to 1. This allows an easy live view on the state of the complete system and its components, individual problems and their possible causes.

**Considered Safety Requirements**

The following requirements have been regarded by measures and components presented in this chapter, which are important aspects for safe adhesion and locomotion in a vertical environment:

**Robust controllers** As given in requirement 3, the closed-loop controllers have to be precise, fast, robust and oscillation-free to allow a safe robot adhesion. This aspect is more or less inherited from the previous work, since the usage of behaviors does not change the internal control algorithm of the chamber controllers.

**Identification of problems** Requirement 4 is considered in terms of the transfer from classic closed-loop controllers to a behavior-based network of single components. The usage of standardized interfaces and meta data allows the analysis of upcoming problems via special tools like the mca browser.

**Balancing of downforces** The uniform distribution of downforces on all three wheels is a main demand to avoid robot tilt as described in requirement 12. By the way, this also helps to reduce wheel slip (requirement 7) and therefore enhances robot navigation, reduces the chance of robot slip and increases the lifetime of the wheels due to lower wear. The presented behavioral network allows this downforce balancing via different closed-loop control and calculation behaviors.

**Shut off of leak chambers** As demanded in requirement 13, the cut off of leak chambers is important to keep the robot adhered to the surface even in cases of large defects. The presented control network solves this task via special deactivation elements and different triggers to reduce the affect of chambers with a high leakage to a minimum.

**Optimal chamber reintegration** Besides the deactivation also the reintegration has to be developed in a way, that the influence of the reintegration on the remaining adhesion system is comparably low. Requirement 14 therefore is handled by different mechanisms like a fast reintegration without testing or a cancelation of the testing phase, if the leakages are too high.

All in all, the presented measures and control elements support robot safety regarding the navigation on a vertical wall in a significant way. As shown in the experiments, the adhesion system is able to keep up higher forces and negative pressures in critical situations, achieve a better and faster distribution of chamber pressures and can be analyzed more easily via some tools. Especially the enhanced force balancing including the corresponding behaviors, but also the deactivation of chambers enhance not only adhesion, but also navigation of the robot. In fact, the adhesion forces are distributed equally among the three wheels, which optimizes the wheel grip (reduces wheel slip and wheel abrasion) and
enhances the motion capabilities of the system. The realization of the control components in terms of behaviors and a complex network allow to use meta data as additional abstract information about the system state, to allow an easy extension of the control system, to facilitate an online analysis of the components, and to detect problems or malfunctions easily. Nevertheless, the system can still fail in a couple of situations. The next step is now to analyze the current system state online to predict upcoming hazards. This aspect will be introduced and described in the next chapter, which deals with the determination of key features, their interpretation and the resulting risk prediction.
8. Online-Analysis of Safety

The previous chapters presented several methods to enhance robot navigation safety in this particular setup. As pointed out, the climbing robot is now able to drive faster on a vertical concrete wall with less wheel slippage and with more balanced adhesion forces compared to the initial state without the shown measures like TCS, SFC and advanced behavior-based adhesion control.

All these measures help increasing safety, but are not able to avoid a robot drop-off in certain situations. If the closed loop pressure and force controllers reach their limits, they won’t be able to ensure the safe adhesion of the robot. Another limitation is based on the perception of the environment. Although the robot is equipped with a light-weighted Hokuyo laser ranger for obstacle avoidance as shown in section 9.1 of the upcoming chapter, these external sensor data have a relatively low accuracy compared to the micro-features which need to be detected for a foresighted evaluation of the surface structure. Therefore, the current system state has to be analyzed to detect safety-critical situations and to predict an upcoming drop-off, if no counteractive measures are triggered. Since external sensors cannot be applied due to the limited payload and mounting space, missing accuracy of standard sensors, and limitations in sensing time or detection window of more advanced sensors, internal values like the chamber pressures have to be taken into account. The main challenge is to determine such key features, which can be used to indicate hazardous system states. Unfortunately, there neither exists a single indicator for denoting a forthcoming drop-off nor the knowledge, which values can be used for the prediction or which portion of influence these values have.

This chapter will deal with these aspects and first present the general concept of risk prediction in section 8.1 and the considered values used for indication. Since there exists a large number of these key features, an optimization method is needed to find the best values and their importance on the prediction (section 8.2). The complete optimization procedure is done via a learning method as shown in section 8.3. This section presents elementary and special steps for determining the needed key features and their influence. Finally, section 8.4 presents experimental results including both, the optimization algorithm and the final risk prediction.
8.1 General Concept

The idea of the online safety-analysis is to make use of the behavior-based adhesion control network or – more precisely – of the meta data of its behaviors as presented previously in chapter 7. The general concept is shown in figure 8.1, which abstracts from any functionality. Common robot control structures use different control levels reaching from the hardware and corresponding electronic boards to closed-loop controllers and function elements up to high-level planners. The idea is to add an evaluation element (blue block) which collects the behavioral data (dashed lines) of the control network (green block) which analyzes the current state of the system and delivers information to additional elements. The surrounding robot software structure with its control and sensor data (solid lines) as well as the behavior network are not affected since the evaluation element is only a surveillant, which triggers existing or additional control elements on top.

![Diagram](image)

**Figure 8.1:** The basic concept of the approach uses behavioral meta data (dashed lines) independent from the surrounding control structure or any functionality.

In this case, the monitored behavior network is of course the adhesion control group with interacting behaviors as depicted in figure 7.16 in previous section 7.5 more detailed. Each behavior contains a controlling component, which is responsible for one part of the adhesion components (e.g. one adhesion chamber), or higher functions. They deliver uniform meta data showing their degree of activeness (activity) and dissatisfaction with

---

1See appendix A.2 for descriptions of the hardware symbols.
the current situation (target rating) as introduced in section 2.2.1. Again, the activity represents the amount of action the behavior wants to perform inside of the network, whereas the target rating is a kind of satisfaction value and shows, how pleased the behavior is in the current situation. The activation values, which are related to the maximal amount of action that can be performed, will not be considered here since they do not represent the real amount of action. A behavior could be stimulated and inhibited in any order and still does not become active if its situation of action does not occur. Based on these meta data, the behavior evaluation is used for risk prediction to detect hazardous situations and to trigger counteractive measures.

8.1.1 Evaluation Function

The question that raises is: What kind of function can be used for the evaluation of the behavioral meta data? Of course, it has to be in a way generic to be able to adapt to a large bandwidth of applications. On the other hand, it has to be customized to the current problem. Although each behavior provides an intuitive value of dissatisfaction and activity, this does not lead necessarily to a meaningful value for the complete system. Therefore, an evaluation function is needed, which takes all meta data from the behaviors and calculates a value representing the needed attribute. The idea is to use this meta data as basic representatives for the network’s state. Additionally, some derived values of the activity and target rating are used which potentially could be important for the evaluation. In the present case, this could be e.g. the average \( \text{avg}(x) \) over a certain period of time, variance \( \text{var}(x) \), difference \( \text{diff}(x) \), lowpass filtered values \( s(x) \) and further derived values. For generality, the presented evaluation methods use a general input vector \( \vec{x} \). In the range of the considered approach, this vector is a concatenation of the considered behavior or derived values with \( x_i \in \{a_j, \text{avg}(a_j), \text{var}(a_j), ... r_k, \text{avg}(r_k), ... \} \) and \( a_j \in \vec{a}, r_k \in \vec{r} \).

The applied method for evaluation is the weighted sum \( F_E(\vec{x}) : [0,1]^n \mapsto \mathbb{R} \). This function is based on the \( n \) meta data and derived values of the behaviors according to equation (8.1). The used weights \( w_j \in \mathbb{R} \) – one for each value – can be chosen arbitrarily.

\[
F_E(\vec{x}) = \sum_{i=1}^{n} (w_i \cdot x_i) \quad (8.1)
\]

The advantage of this method is that it is able to consider all important values and can generate a result in any desired range. Of course, the main problem is to determine suitable weights which produce the designated functionality. In fact, it is impossible to identify these weights by hand so learning or optimization techniques have to be used. Nevertheless, it is still unclear if this function is suitable for evaluation or if a more complex function is needed. Appendix A.1.5 sums up further functions like maximum or weighted mean, which could be used instead of the weighted sum. Of course, it has to be kept in mind that one of the more generic evaluation functions might become necessary, if the weighted sum does not provide the desired results. But, based on the different characteristics, advantages, and disadvantages of these functions, the weighted sum seems to be a good compromise between effort and possible results.

This presented approach is only possible because of the redundant adhesion chambers. Since robot adhesion is ensured by a couple of chambers, some of them may fail for a
short period of time without endangering the system. In practice, the front chambers in driving direction are exposed to hazardous features first which allows a judgement of the upcoming terrain. First experiments have proven that pressure values itself are not sufficient for risk prediction since they do not consider some important aspects like the value and point of downforce, leakages, control differences and much more. Therefore, virtual sensor values in the form of activity and target rating are evaluated instead of the real sensors as mentioned before. Especially the different target rating values provide information about the state of the adhesion system because they represent individual degrees of satisfaction of controller elements under the implicit assumption that functions $f_a$ and $f_r$ (see section 2.2.1) are chosen properly for each single behavior.

### 8.1.2 Risk Prediction

In the current approach, the evaluation function $F_E(\vec{a}, \vec{r})$ uses the behavioral values of the adhesion control network to analyze the current safety state of the climbing robot to get the risk the robot actually is exposed to. The desired characteristics of this estimation is to receive a risk value, which is one or above if the robot will fail some time before it happens, if no evasive action is performed. On the other hand, it must stay below one if the robot adhesion is not endangered to avoid false positives. Finally, the risk value should indicate a potential drop-off early enough to allow counteractive measures like driving back to a safe position or an inflation of the adaptive sealings to enhance leak-tightness so that the adhesion system can recover.

$$E = F_E(\vec{a}, \vec{r}) = \sum_{i=1}^{n} \left( w_{a_i} \cdot a_i + w_{sa_i} \cdot s(a_i) + w_{r_i} \cdot r_i + w_{sr_i} \cdot s(r_i) \right)$$  \hspace{1cm} (8.2)

As given in equation (8.2), activity and target rating values $a_i$ respective $r_i$ of behavior $i$ are used in combination with corresponding weights $w_{a_i}$ and $w_{r_i}$. Again, the activity provides information about the action of a behavior which can be e.g. the valve position of an adhesion chamber. The target rating gives information about the satisfaction of a component which might be the difference of current and desired control value or an estimation of chamber leakages. Both values are important for evaluation since the combination of them allows a conclusion on the overall behavior state. If a behavior is unsatisfied, but not active, it does not have the chance to change its situation and therefore is not critical. In contrast to that it might be more problematic if it is active, but still unsatisfied, so it is not able the reach the desired goal (e.g. a desired control value).

In addition to the two behavior values, also lowpass filtered meta values $s(a_i)$ and $s(r_i)$ with according weights are taken into account. It would also be possible to calculate variations of these values like average or median as shown before and to use these additional information in the same way. Recent experiments have shown that these values may allow a better prediction, but this enhancement comes with two restrictions: First, more weights have to be determined. Second, there is a higher specialization to certain situations and time-dependent parameters like vehicle velocity. Therefore, only the current and lowpass filtered values as calculated via equation (8.3) are used here with filter result $s_{t-1}(x)$ of the previous cycle.
8.2 Optimization Problem

\[ s_t(x) = 0.3 \cdot x + 0.7 \cdot s_{t-1}(x) \quad \text{with } s_0(x) = 0.0 \] (8.3)

The evaluation process is depicted in figure 8.2. The used weights define the final evaluation function and have to be determined for the given application as well as a selection of variations of the applied behaviors since not all values are leading to the goal. In the present case, the weights are limited to a range of \([-1, 1]\) which seems to be suitable for the given problem. Nevertheless, it is possible to adapt this range if it becomes necessary throughout the optimization or prediction process, if the results are insufficient.

**Figure 8.2:** The generic evaluation function, as it is used within the context of this thesis, based on behavior meta data \(a\) and \(r\) and lowpass filtered variations of them.

8.2 Optimization Problem

It should be obvious, that the evaluation function delivers the desired results only if the correct weights are used. The remaining problem is now the selection of those values, which potentially could be important for risk estimation and the determination of the corresponding weights. In general, the selection of key features can be neglected since the importance can be given by a small weight value equal or close to zero. Nevertheless, the search space increases tremendously with the number of used behavior values. Assuming, that each weight is in the range of \([-1, 1]\) with discrete steps of 0.01, the number of possibilities is \(200^{2^n \cdot w}\) with the number of behaviors \(n\) and the number of weights \(w\) needed for one meta value including its variations. Additionally, each behavior may provide both meta values activity and target rating. Even under the presupposition of a very small network of only five behaviors \((n = 5)\) and two variations per value \((w = 2)\), the total number of 20 weights leads to \(200^{2^5} = 200^{20} \approx 10^{46}\) possibilities of choosing these weights.
This large number of possibilities for the weights cannot be set by hand. The available approach uses a total number of 47 behaviors which makes the use of an automatic procedure mandatory to determine the needed values. But, the sequential testing of all possibilities would take too long. Obvious solutions are learning methods which can be applied if no or only insufficient knowledge of the system is given. Independent from the approach, each learning method needs some kind of training data as well as a rating function which has to be optimized by the algorithm. The forthcoming paragraphs will show a scale unit for the degree of safety as well as the training data, which can be recorded from simulated or real-world scenarios and present possible learning methods to find suitable weights.

8.2.1 Degree of Current Adhesion

The degree of current adhesion is an important measuring unit to determine, how safe the robot is adhered to the wall. This value including the identification of a drop-off can be calculated via two indicators: The degree of current downforce and the point of downforce. In the context of this thesis, an adhesion score $S_A$ with calculation function $F_{S_A}(F_{D\|z}; x_F^{act}, y_F^{act}) : \mathbb{R}^3 \mapsto [0, 1]$ has been developed which uses both indicators as given in equation (8.4):

$$S_A = F_{S_A} \left( F_{D\|z}; x_F^{act}, y_F^{act} \right) = \max \left( F_{S_A|F}(F_{D\|z}), F_{S_A|P}(x_F^{act}, y_F^{act}) \right) \quad (8.4)$$

The first indicator, the current downforce value $F_{D\|z}^{act}$, can be derived directly from the measurements by the embedded load cells as given in equation (7.29). The score of downforce $F_{S_A|F}$ is calculated according to equation (8.5) with lower and upper thresholds $\hat{F}_{min}^{AF}$ and $\hat{F}_{max}^{AF}$ and the limit function according to the notation in appendix A.1.2. If $F_{D\|z}^{act}$ is below the lower threshold the robot falls down since the adhesion force in combination with the friction of the drive system does not generate enough counterforce to gravity. The second threshold value $\hat{F}_{max}^{AF}$ determines a downforce value which ensures adhesion. Both threshold values depend on system parameters like weight distribution or friction coefficient. Figure 8.3a illustrates the score function.

$$F_{S_A|F}(F_{D\|z}) = 1 - \left( \frac{F_{D\|z}^{act} - \hat{F}_{min}^{AF}}{\hat{F}_{max}^{AF} - \hat{F}_{min}^{AF}} \right)^0 \quad (8.5)$$

The second score function $F_{S_A|P}$ depends on the point of downforce and describes the chance of robot tilt according to equation (8.6). If the distance between the current downforce point $\vec{P}_{D\|z}$ (at coordinates $x_F^{act}$ and $y_F^{act}$) and the robot center lies above threshold $\hat{l}_{max}^{AF}$ the robot drops off. Figure 8.3b shows this function exemplarily. In general, this threshold is equal to $\frac{1}{2}l_W$ which is the radius of the stability circle as illustrated in figure 3.14 with distance $l_W$ from wheel to robot center. If this value is exceeded, one wheel is lifted and the robot drops off. Nevertheless, it is possible to chose a smaller threshold to keep up a safety range.
Figure 8.3: Illustration of the adhesion score functions depending on the current downforce (a) and its position (b).

\[
F_{S_A/F}(x_{FD}^\text{act}, y_{FD}^\text{act}) = \left\langle \frac{\sqrt{(x_{FD}^\text{act})^2 + (y_{FD}^\text{act})^2}}{l_{AP}^\text{max}} \right\rangle_0
\]  

(8.6)

It has to be pointed out, that the resulting value is a current rating of the adhesion in a way, that the robot drops off if \( S_A \) reaches one\(^2\). Of course, it is possible to detect a drop-off more easily, e.g. if the wheel’s downforces \( F_{W_{ij}^z}^\text{act} \) go down to zero. But, it is helpful to know, if the system is approaching that critical state or if it is adhered safely.

### 8.2.2 Training Data

Independent from the type of learning algorithm some experimental data for training the desired functionality are needed. Since the behavior evaluation is based on behavioral meta data, a logging mechanism was set up to store all these values as well as the corresponding adhesion score \( S_A \) at each time step. Therefore, each training set contains meta data and score over time \( t \in \mathbb{N} \). For the existing risk prediction method, the original and filtered behavior values and the adhesion score is used as shown in table 8.1. Of course, the size \( m \) of the tables varies from one dataset to another whereas the setup of behaviors and variations has to be fixed.

Table 8.1: Example table of a training set for risk prediction. The general setup for behavior evaluation are behavioral meta data and score values.

<table>
<thead>
<tr>
<th>time</th>
<th>behavioral meta data and variations</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t )</td>
<td>( a_1(t) )</td>
<td>( s(a_1(t)) )</td>
</tr>
<tr>
<td>0</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>1</td>
<td>0.90</td>
<td>0.99</td>
</tr>
<tr>
<td>2</td>
<td>0.90</td>
<td>0.98</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( m - 1 )</td>
<td>0.00</td>
<td>0.01</td>
</tr>
</tbody>
</table>

\(^2\)In fact, it is not hundred percent sure that the robot really drops off, if the adhesion score reaches a value of one. It can still remain adhered due to certain influences like robot orientation (compare section 3.4.7) or the current friction coefficient. So the chosen function still keeps up a certain safety range
As training examples, different situations have to be considered to find weights, which work in a broad number of cases. Therefore, the robot has to be faced with situations in which the adhesion system reaches its limits and the robot falls down \((S_A = 1)\) as well as situations which are harmless or still manageable by the system \((S_A < 1)\). In simulation, the generation of training data is much more simple since the robot hardware cannot be destroyed in case of a drop-off. If the real system is used, it has to be secured to control the drop-off.

### 8.2.3 Learning Methods

One remaining question is now: What kind of method can be used to retrieve the desired functionality? As mentioned before, the only suitable way to determine the large number of weights are automatic methods. In this case learning approaches would be appropriate solutions to handle this large amount of variables. In literature, different approaches exist which can be used for learning in general and to find the optimal weight values based on the recorded training data. A comprehensive overview on machine learning and corresponding methods and algorithms can be found in several books, e.g. in [Alpaydin2004, MacKay2003, Thrun1997].

A very common approach in the field of machine intelligence are artificial neural networks (NN). These biological inspired methods are widely spread among different scientific disciplines since they meet a manifold number of problems like pattern recognition, clustering or optimization tasks [Jain1996]. The learning process itself results in continuous updates of the neural network structure and of the neuron weights. In general, one distinguishes three main classes of learning procedures via NN: Supervised, unsupervised and hybrid learning. Supervised learning needs training data consisting of input samples in combination with the correct answers which enables it for classic pattern recognition tasks. In contrast to that does unsupervised learning not require the correct output. This method tries to find correlations between the training sets which makes it applicable e.g. for clustering or data analysis tasks. Hybrid learning combines both aspects by learning some weights with supervisor and some without.

So far, artificial neural networks seems to be a perfect method to solve the given problem since they learn weights internally to perform the desired task. But, the disadvantage of artificial neural networks – related to the existing challenge – can be summarized in a quote by Harmon, who gives a flight control system as a negative example: “Simple neural networks can’t learn to fly the plane unless there is a set of known answers, so if we don’t know how to build controller in the first place, simple supervised learning won’t help” [Harmon1996, page 2]. In the present case, exactly these correct answers cannot be given since it is not known, in which way the input data (activity and target rating) are related to robot’s forthcoming adhesion situation. The problem is based on the fact, that e.g. one meta data set needs to stay below 1 in one case but might notice an upcoming drop-off in another case. One can identify false-positives certainly for non-hazardous training sets. The clear detection of required evaluation values \(E \geq 1\) is not possible because it is not clear if other meta data exist in the desired time range for risk prediction, which already notice an upcoming drop-off with the current weights, or not. Also unsupervised learning will not help in the given situation since it just allows a clustering of the behavior data without any further processing or background knowledge.
A similar approach is reinforcement learning (RL). It combines the disciplines of dynamic programming and supervised learning, which enables it to optimize a given problem via trial and error [Harmon1996] like special closed-loop controllers [Steiner2009] or the actions of multi agent systems [Kok2006]. As in the case of NN, reinforcement learning also makes use of input-output samples to learn the desired function – but without knowing the correct answers. Instead of this it operates on a reward function, which rates the current results or state of the system. In general, a RL system consists of three elements: The dynamic environment is the interaction partner of the RL system. The interaction between both parts takes place in the form of actions triggered by the learning system and states delivered from the environment. A reinforcement function describes the goal of the learning system as a kind of mapping from state-action pairs to reinforcement values. In general, the RL system receives a reward if the chosen action leads to the goal which should be achieved. Nevertheless, it is also possible to inflict a penalty, if the goal cannot be reached by the current action or if the current state should be avoided. The value function tries to determine the optimal way to reach the goal. This is done with the help of a policy determining which action should be executed in each state. Additionally, each state is associated to a value based on the sum of reinforcements from that state to the final one. Different characteristics and methods of RL systems can be found in [Sutton1998] and [Kaelbling1996].

In general, reinforcement learning has similar limitations as the classic artificial neural networks. In contrast to training examples with the correct answers, RL methods need a reward function depending on the current state of the system. Nevertheless, RL is more linked to a mapping from situations to actions than for the given problem of retrieving weights for risk prediction. One could imagine to use reinforcement learning for adapting robot navigation including emergency actions in hazardous situations, but RL needs – depending on the size of the state space – a certain time of interaction between learning system and environment. This large number of trials cannot be performed by the real robot in the needed amount because of time constraints and hardware limitations.

Other methodologies try to approximate an optimal solution via heuristic optimization like in the case of simulated annealing (SA) as described by [Kirkpatrick1983]. Here, a non-optimal solution is replaced by a neighbor solution with a certain probability depending on the results of the cost function of both solutions and on a nonincreasing cooling function. This temperature value changes the update policy over time from nearly random to optimize only. The analogy and eponym of this mechanism can be found in metallurgy, in which atoms should arrange in an optimal way. At the begin of the process, the material is fluid and has a large freedom to move. Related to the algorithm this allows e.g. the usage of worse solutions which is important to escape from local minima. Over time, the material cools down which limits the motion capabilities until the material is solid. The algorithm reduces the probability of taking worse solutions in each temperature step until they are nearly ignored.

The approach of simulated annealing is able to approximate to a global optimum depending on a given function even in a large search space as it is given here. Nevertheless, it does not seem the best idea to follow and to optimize only one solution in the present case as it is also done in similar methods like random walk [Lawler2010]. Since the internal nature and correlation of the search space and the characteristics of optimal solutions is not clear, a method should be preferred, which takes a couple of solutions into account.
This leads to another common technique to determine the needed weights. In the present case, the principle of **genetic algorithms** (GA) is applied to learn the best weights as it will be presented in upcoming section 8.3. It might be possible, that other methods like simulated annealing work faster, but the speed of the algorithm is not that important since the weights will be trained offline.

### 8.3 Learning via Evolutionary Techniques

Genetic algorithms are a widespread method in terms of optimization problems. The goal is to get a result out of the complete solution space, which maximizes or minimizes a certain function. In the present case, a genetic algorithm has been developed which updates the evaluation weights randomly until the desired performance is achieved. The next paragraphs will briefly introduce the basic principle of genetic algorithms (section 8.3.1) and applications (section 8.3.2). In section 8.3.3 the concept of the developed learning method will be given. Upcoming sections will describe the aspects of the genetic algorithm in detail. Especially the identification of *good* weights will be discussed.

#### 8.3.1 Introduction to Genetic Algorithms

Genetic algorithms are a biologically inspired optimization method which imitates evolution in nature [Gerdes2004]. In general, there exists a population $P(s)$ at time or evolution step $s$ which consists of individuals. Each *individual* (also known as a *chromosome*) $c$ is one possible solution for the given problem and located inside of the search space $S$. The search space (also known as *alphabet*) has to be selected carefully: If it is too large the computational amount increases, on the other hand good solutions might be excluded if the alphabet is too small. The first population $P(0)$ consists of random individuals and – if possible – of already known good individuals. Research of genetic algorithms started with binary patterns as an alphabet, but nowadays also integer or floating point numbers are very common as it is applied in the scope of this thesis in terms of the weights.

The goal function of the optimization process is encoded as a *fitness function* $F(c) : S \mapsto \mathbb{R}$ which influences the chance of an individual to survive. The fitness determines, if an individual is a good solution (compared to the other individuals) or not. During an evolution step, the fitness of each individual is calculated. The probability of an individual $p(c)$ to survive generally depends on the individual’s fitness $F(c)$ and on the population’s fitness, which is the sum of fitness values of all individuals (equation (8.7)).

$$p(c) = \frac{F(c)}{\sum_{j \in P} F(j)} \quad (8.7)$$

An individual $c_i \in P$ is selected for the new generation, if equation (8.8) is satisfied by the *Roulette wheel method* as described by [Gerdes2004]. This process corresponds to the *natural selection* of individuals with random variable $r \in [0, 1]$.

$$\sum_{j=0}^{j<i} p(c_j) \leq r < \sum_{j=0}^{j<i} p(c_j) \quad (8.8)$$
The result is an intermediate population $P'(s)$ consisting of the survivors. The number of individuals $|P(s)| = |P'(s)| = |P(s + 1)|$ stays constant throughout the complete process. The survivors are then adapted by genetic operations. Via \textit{mutation}, single genes inside of an individual are changed with a certain probability. In case of binaries, this mutation is a switch from zero to one or vice-versa (figure 8.4). In nature, mutations e.g. can be caused by mutagenic substances or radioactivity. Another operation is the so-called \textit{crossover} which merges genes of two chromosomes. Depending on the used type of crossover this can be a row (one-point-crossover) as shown in figure 8.4, single genes or other options. In cases of non-binary genes, these two operations adapt the values e.g. via random offsets or interchange complete integer or floating point numbers.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure8.4.png}
\caption{Basic genetic operations which change the chromosome.}
\end{figure}

One evolution step, starting with the current population $P(s)$ at step $s$, can be summed up as follows:

1. \textbf{Fitness calculation:} Determine fitness $F(c)$ of all individuals $c \in P(s)$ of the current population (parents).

2. \textbf{Breakup criterion:} Check if the current population $P(s)$ or an individual inside of it already meets desired criteria and cancel the optimization process if this is the case.

3. \textbf{Selection:} If further improvements are necessary, randomly select individuals with their corresponding probability $p(c)$ for the intermediate population $P'(s)$ (survivors).

4. \textbf{Genetic operations:} Adapt all survivors $c \in P'(s)$ via genetic operators \textit{mutation} and \textit{crossover} with certain probabilities.

5. \textbf{New population:} Set updated $P'(s)$ as new generation $P(s + 1)$ and continue evolution with the new child population.

Figure 8.5 illustrates one evolution step related to the current approach using training data and the adhesion score for comparison. A more detailed introduction to genetic algorithms can be found in several publications [Gerdes2004, Rudolph1994, Whitley1994].
8.3.2 Related Work in Genetic Approaches

In literature, multiple variations of genetic algorithms and a countless number of applications exist. Some researchers use genetic algorithms to train weights of a dissimilarity function to identify trademarks [Chan1999] or to optimize network performance in the range of the open shortest path first weight setting (OSPFWS) problem [Ericsson2002]. A similar combinatorial issue is the multiple container packing problem (MCPP) which can also be handled by a weight-coded GA [Raidl1999].

For the present application, the focus lies on interesting research activities related to robot motion and on approaches, which also use genetic algorithms for weight learning. Koos and Mouret for instance presented one application of genetic algorithms in the range of robotic locomotion [Koos2011]. They use a multi-objective evolutionary algorithm (NSGA-II [Deb2002]) to discover suitable locomotion modes of a hybrid mobile robot for the current terrain characteristics. The robot itself is a so-called wheel-legged system which is equipped with four legs with 2 DOF each and driven wheels as feet. Based on this setup, the authors try to adapt three parameters which characterize different types of motion for flat ground, grass-like terrain and tunnels. The parameters are 1) wheel speed, 2) amplitude of leg movement and 3) robot posture, which have to be adapted depending on the terrain. In operation, the robot tries different parameter sets and measures its effort related to four objectives: The average vehicle speed, the expended energy of the chosen locomotion type, a transferability measure of the terrain and behavioral diversity. In their implementation, the authors used a population of 40 individuals (parameter sets). The evolution process takes place in simulation based on an approximated transferability value of the terrain. Every 10 generations the parameters of the individual with the highest behavioral diversity are taken and executed on the real robot. The result of this test is used to enhance the approximation of terrain transferability. In total, the system performs 200 evolution steps (in simulation) including 20 experiments on the real system. The main drawback of this application is the question, if evolutionary algorithms are really needed to determine this limited set of parameters. Maybe it would be better to use additional world knowledge to enhance the determination of suitable parameters. Since the influence of the terrain on the motion capabilities can be described well, the system only has to test a small number of predefined parameter sets, which are suitable to different surfaces and use the best one or adaptations of it.

Another application for genetic algorithms has been introduced by Burchardt and Salomon in the range of path planning algorithms for a team of small-sized RoboCup\textsuperscript{3} robots [Burchardt2006]. Their goal is to find an optimal collision free path between two points. They encoded the path in form of up to three way points between the start and end position as integer x and y coordinates. Additionally, also the number of intermediate points is part of a chromosome so the number of way points varies from zero to three. To calculate the individual’s fitness the path length is considered as well as a term of collision avoidance. Equation (8.9) describes the used error function with the optimization element concerning the two criteria. Here, $|\text{path}_i|$ denotes the length of the path between points $\vec{p}_{i-1}$ and $\vec{p}_i$. The collision penalty depends on the number of collisions $n_{\text{collision}}$, a fix obstacle radius $r_o$, the distance $d_i$ between robot path and obstacle center and a constant penalty $c_{\text{penalty}}$ (set to twice the length of the playing field).

\footnote{\texttt{http://www.robocup.de/}}
8.3. Learning via Evolutionary Techniques

$$f = \frac{4}{\sum_{i=1}^{n_{\text{path}}} |\text{path}_i|} + \frac{n_{\text{collision}}}{\sum_{i=0}^{\text{collision}} (c_{\text{penalty}} \cdot \max(0, r_o - d_i))}$$

(8.9)

Of course, the fitness of an individual increases if this error function becomes lower. As genetic operators, a set of six different mutations are used which are suitable to the field of path planning. It includes simple genetic modifications, but also the insertion or removal of way points (which changes the length of a chromosome). For evolution itself, the \((\mu + \lambda)\)-strategy [Brownlee2011] is used, which selects the best \(\mu\) parents and best \(\lambda\) offsprings for the next generation. By this elitist mechanism the surviving of the best individuals is guaranteed. Finally, the developed path planning implementation met real-time constraints and worked even on the limited micro controllers as they are used in the RoboCup’s small-size league. This approach is very interesting since it is designed for an optimum of result and calculation costs (time and memory). But, in the present case the training phase is performed offline so the convergence speed is less important than the final result. Nevertheless, the error function which is used for fitness calculation might be interesting also for the present application.

In the range of scoring functions Flom and Robinson made use of genetic algorithms to determine the current state of a Tetris game and to find the best position for the current brick [Flom2004]. It has to be mentioned that solving Tetris is \(NP\)-complete and therefore still a research topic in computer sciences [Demaine2004]. The interesting point is here, that the authors also use a weighted sum of features as evaluation function and have to determine weights to optimize it. In this case, they evaluate five numerical features: 1) the height of the tallest column (pile-height), 2) the number of closed holes which are covered by bricks on top, 3) the number of deep and narrow spaces (wells), 4) the number of lines made by the playing agent and 5) a measure of bumpiness of the pile’s top. Depending on the evaluation of these five parameters the agent should decide where to place the current block for an optimal play. The authors used two indicators \textit{number of lines} and \textit{ratio of lines per piece} of a game for determining the fitness. As chromosomes, an array of floating point numbers (weights) in a binary representation has been chosen, which are adapted via an adaption of the \textit{Genitor algorithm} [Whitley1989]. The key features of a Genitor algorithm are the explicit use of ranking of individuals and the abandon from the generation idea (parents and children can co-exist). In the available approach, two individuals (parents) are selected randomly to create one child via a crossover operation. Additionally, a twin (copy of the child) is mutated randomly. Both are added to the population and the two worst individuals depending on the fitness value are removed. The authors have proven that it is possible to train weights for an evaluation function which enables an agent to play Tetris. They point out, that there is no obvious relationship between the different weights since there exist different ways to play the game. In fact, this work is a good example for the possibilities of genetic algorithms, although the authors did not describe why a Genitor based algorithm has been used. In fact, it appears to converge faster than other genetic algorithms. But, in the scope of the present thesis the Genitor algorithm is not the best solution since “the algorithm is less prone to wander in the search space” [Whitley1989, page 121].
In fact, the combination of safety analysis using meta data from a behavior-based robot control network and genetic algorithms for determination the evaluation function has not been applied so far and will be described more detailed in the further sections.

### 8.3.3 Genetic Algorithm for Optimization

In the present case, the desired weights are determined via a genetic algorithm. As usual, a population $P(s)$ of individuals at step $s$ is needed. Each individual has a chromosome $c$ with genes which can mutate randomly in a predefined way to optimize the desired function. In this case, one individual $j$ consists of a vector of weights $\bar{w}_j = (w_{j|a_1}, w_{j|a_1}, w_{j|a_1}, w_{j|a_1}, ..., w_{j|a_1})$ which is used to calculate the weighted sum of $n$ behaviors\(^4\) as shown in equation (8.2).

Figure 8.5 depicts the concept of the learning procedure using the genetic algorithm. The data fundament of the learning algorithm is the set of training data which has to be recorded beforehand. As given in section 8.2.2, each training set is a list of behavior values, variations of them and an adhesion score over time. The score data $S_A(t)$ represents the current threat at each time step $t$ and depends on the current value of downforce and its distance from the robot center (equation (8.4)). It is zero, if the robot is adhered safely and one if it lost adhesion and falls down.

At the beginning of the optimization process, a random population $P(0)$ of individuals is generated. Each individual is created with random genes in form of weights. Afterwards, the following four steps are executed once per iteration step $s$:

1. For each individual (set of weights) inside of population $P(s)$, the training sets are used to calculate the evaluation data $E$ over time according to equation(8.2), which is the weighted sum of the behavior values.

2. Afterwards, the fitness $F(c)$ of each chromosome (individual) is calculated based on the resulting curves, which are compared to a desired characteristic depending on the adhesion score $S_A$. One characteristic e. g. can be that the evaluation value should exceed a certain limit one second before the adhesion score is at maximum. This would give the robot a reaction time of one second to start counteractive measures to prevent a drop-off. Depending on this comparison the fitness of each individual is set and used for a selection of individuals $P'(s)$ for the next generation.

3. This generation is created by a random selection of individuals depending on their fitness. In this case, an individual is fit if its weights allow a good prediction of the adhesion rating. The fittest individuals have the greatest chance to survive and will probably be selected multiple times so that the number of elements in the new population remains constant.

4. Finally, each gene (weight) of every chromosome has a certain probability $p_m$ for mutation. In this context, a mutation can be e. g. a weight update via a random offset or similar operations. The result is an updated children population $P(s+1)$ generated out of the parent population.

\(^4\)The final implementation allows also the omission of certain activity, target rating or filtered values to reduce the search space, e. g. if a value is set fixed and does not contribute anything to the evaluation function.
8.3. Learning via Evolutionary Techniques

Individual $j$:

$$\vec{w}_j = (w_{j|a_1}, w_{j|sa_1}, \ldots, w_{j|r_n}, w_{j|sr_n})$$

Population $P(s)$:

Evaluation of individuals $E(t)$ (section 8.1.2)

Training sets (section 8.2.2)

Adhesion $S_A(t)$ (section 8.2.1)

Comparison (section 8.3.4)

Selection (section 8.3.5)

Population $P(s+1)$

Mutate (section 8.3.6)

Survivors $P'(s)$

Figure 8.5: Concept of the genetic learning algorithm: Each individual is a set of weights which is evaluated with given behavior values, compared to a desired function, selected and updated.

The complete process has to be repeated several times until at least the best individual reaches the desired characteristics. These characteristics – which are needed for the identification of good weights – are the basis of the fitness function to select the best individuals and will be explained in the forthcoming section.

8.3.4 Rating of Individuals

Of course, the setup of the fitness function is the most difficult part since it has to be set properly to identify good weights out of the large number of possibilities to achieve the desired performance. Therefore, a kind of rating function is needed to detect suitable sets of weights. In the present case, an individual is good if the evaluation function $F_E$ from equation (8.2) in combination with the individual’s weights is a proper prediction of the recorded adhesion score $S_A$. This proper prediction is described in terms of a rating function $R_E$. At first, the weights of one individual are used to calculate the evaluation value $E$ according to equation (8.2) for each timestep $t$ of one training set (compare section 8.2.2). The result is a list of evaluation values $E(t)$ over time, which have to be
compared to the corresponding adhesion scores $S_A(t)$ of that data set. For comparison, some requirements must be fulfilled:

- In general, the evaluation value should stay below the adhesion score to avoid false-positives.
- If the rating shows that the robot drops off ($S_A(t) = 1$) the evaluation value must be high a certain time $\Delta t$ earlier – but not too early.

This time value $\Delta t$ determines the minimum reaction time which is needed for the robot to start and execute counteractive measures to avoid the drop-off. In the present case it is assumed, that a time range of one to two seconds would be optimal for reaction, so it is set to $\Delta t = 1$ s. The rating of weight evaluation $R_E$ is done according to equation (8.10) and needs to be calculated for each combination of individual and training set. It follows the same idea as the error function used by Burchardt et al. as described in section 8.3.2.

$$
R_E = - \sum_{t=0}^{m-1} P_{E\Delta}(t) \right \{ \sum_{t=0}^{k} P_{E\text{unw}}(t) - M_{E\text{unw}} \right \} \text{general deviation} \quad (8.10)
$$

$$
- \sum_{t=k+1}^{k+\Delta t} P_{E_{\text{req}}}(t) - \sum_{t=k+\Delta t+1}^{m-1} P_{E_{\text{req}}}^{2}(t) - M_{E_{\text{req}}}^{2} \right \} \text{unwanted values} \quad \text{required values}
$$

This rating considers three aspects: A general deviation from the adhesion score, the avoidance of unwanted values and a penalty for missing required values.

**General deviation** At first, the evaluation function $E(t)$ should stay below the adhesion score $S_A(t)$, otherwise the rating value is diminished by a penalty $P_{E\Delta}$ according to equation (8.11). Here, the difference between evaluation and score $(E(t) - S_A(t))$ is calculated and multiplied with the evaluation value again. The concept behind this calculation is that the difference is less severe if the evaluation value is low. Finally – to diminish the influence of small differences – the resulting value is raised by the power of 3.

$$
P_{E\Delta}(t) = \begin{cases} 
(E(t) - S_A(t)) \cdot E(t) \cdot 3, & \text{if } E(t) > S_A(t) \\
0, & \text{else} 
\end{cases} \quad (8.11)
$$

This aspect is illustrated in figure 8.6 on the left side. The optimal case is shown on top (a) with an evaluation value $E(t)$ staying below the adhesion score $S_A(t)$. In contrast to that, equation (8.11) inflict a penalty for too high values as marked in figure 8.6b.
Unwanted values The second aspect is the avoidance of unwanted values which produce false alarms. This is split up into a penalty based on the differences of evaluation and adhesion rating \( P_{E\text{unw}} \) according to equation (8.12) and a basic malus \( M_{E\text{unw}} \), if at least one undesired value exists (equation (8.13)). \( P_{E\text{unw}} \) is calculated similar to \( P_{E\Delta} \) from equation (8.11) since it also multiplies the difference between evaluation value \( E(t) \) and score threshold \( \hat{S}_{\text{haz}} \) with the maximum of evaluation and score value \( S_A(t) \) to reduce the influence of this difference in cases of small values. Equivalent to that the result is also raised by the power of 3. The used threshold \( \hat{S}_{\text{haz}} \) denotes a hazardous situation, which is set to 0.9 in the present case. The constant factors \( 10^9 \) and \( 10^8 \) have been determined experimentally to give a strong penalty in cases of these unwanted values. These and some more factors are important for a balanced rating function which needs to consider false positives and false negatives in the same way. For sure, they have to be set depending on the application and certain conditions. If e.g. the training samples are ten times longer than in the present case, it might become necessary to enlarge these parameters by the same factor for balancing the different penalties.

\[ P_{E\text{unw}}(t) = \begin{cases} \left( (E(t) - \hat{S}_{\text{haz}}) \cdot \max \left( E(t), S_A(t) \right) \right)^3 \cdot 10^9, & \text{if } E(t) > \hat{S}_{\text{haz}} \\ 0, & \text{else} \end{cases} \]  
\[ M_{E\text{unw}} = \begin{cases} 10^8, & \text{if } S_A(t) < \hat{S}_{\text{haz}} \forall t \in [0,k] \land \exists E(t) \geq \hat{S}_{\text{haz}}, t \in [0,k] \\ 0, & \text{else} \end{cases} \]
Examples for these unwanted amplitudes are given in figure 8.6. At bottom left (c) the evaluation of the weights produces two false alarms which reach or exceed the hazard limit although the adhesion score remains in a more or less safe state. Figure 8.6f shows a case in which the adhesion score reaches the limit, but the evaluation value exceeds it too early. The second peak at timestep \( k \) might be tolerable whereas the first one at step \( j \) is not.

The value \( k \) depends on the type of training set: If the adhesion score stayed below \( \bar{S}_A^{\text{haz}} \) the complete dataset is processed here (equation (8.14)). Otherwise it considers only the timespan to the timestep \( t^{\text{haz}} \) at which the adhesion rating reached a hazardous value minus the double time \( \Delta t \). This describes the desired timespan of the reaction time with \( \Delta t \leq t^{\text{react}} \leq 2 \cdot \Delta t \).

\[
k = \begin{cases} 
m - 1 & \text{if } S_A(t) < \bar{S}_A^{\text{haz}} \forall t \in [0, m - 1] \\
t^{\text{haz}} - 2 \cdot \Delta t & \text{else} \end{cases} \quad (8.14)
\]

Figure 8.6 shows the two characteristics of \( k \) with the first case (a-c) in which the adhesion score stayed below the threshold and the second one (d-f) in which \( k \) denotes the begin of the desired prediction phase.

**Required values** As before, a malus and a penalty for missing required values are applied, if the adhesion score \( S_A \) of this data set is above the threshold \( \bar{S}_A^{\text{haz}} \) at least once (so that \( k < m - 1 \)), as illustrated in figure 8.6e. Now, the evaluation \( E(t) \) has to reach a value of 1 within the range \([k + 1, k + \Delta t]\) to predict the drop-off. Otherwise, the malus \( M_{E_1}^{\text{req}} \) from equation (8.17) is added as well as the first part \( P_{E_1}^{\text{req}} \) (equation (8.15)) of the penalty, which is based on the difference of evaluation value to one if it is smaller than this.

\[
P_{E_1}^{\text{req}}(t) = \begin{cases} 
(1 - E(t))^3 \cdot 10^9 & \text{if } (E(t) < 1 \wedge E(t) < 1) \forall t \in [k + 1, k + \Delta t] \\
0 & \text{else} \end{cases} \quad (8.15)
\]

\[
P_{E_2}^{\text{req}}(t) = \begin{cases} 
(1 - E(t))^3 \cdot 10^6 & \text{if } E(t) < 1 \\
0 & \text{else} \end{cases} \quad (8.16)
\]

\[
M_{E_1}^{\text{req}} = \begin{cases} 
10^8 & \text{if } E(t) < 1 \forall t \in [k + 1, k + \Delta t] \\
0 & \text{else} \end{cases} \quad (8.17)
\]

Again, the difference is raised by the power of 3 to reduce the effect of small difference values. In the same way, the constant factors \( 10^9 \) and \( 10^8 \) are set for weightening the different parts of the presented rating function. Here, the same factors are used compared to the previous case of unwanted values in equations (8.12) and (8.13) to consider false positives and false negatives in the same way. The second part \( P_{E_2}^{\text{req}} \) of the penalty – given in equation (8.16) – tries to push the evaluation function above 1 over the remaining time steps from until \( m - 1 \) if the hazardous adhesion limit has been reached. As before, \( 10^6 \) is a constant multiplication factor to give this penalty a stronger influence than the general deviation in the considered range, but
less than in the case of missing required or unwanted values. This value has to be set according to the other factors, which must have a stronger influence than this one, to perform a balanced and goal-oriented rating.

An example for this is illustrated in figure 8.6d: Here, an ideal prediction of the adhesion score is given by the evaluation function \( E(t) \) since the evaluation stayed below the score at the beginning, but reaches a value of 1 inside of the desired time range of \([k + 1, k + \Delta t]\). Only the small downward peak after \( t_{haz} \) is not optimal and will be penalized. In contrast to that, figure 8.6e corresponds to a strong penalty and malus \( M_{E_k} \), since there is no required evaluation value above 1 in the desired range. The raising of \( E(t) \) at the end of the plot comes to late.

If more than one training set is used, the mean-square average of these \( p \) ratings is calculated according to equation (8.18). It has been chosen to give those ratings a stronger influence which have a worse value than others. Since the ratings \( R_{E_i} \) are \( \leq 0 \) the complete rating has to be set negative.

\[
R = -\sqrt{\frac{1}{p} \cdot \sum_{i=0}^{p-1} (R_{E_i}^2)} \quad (8.18)
\]

The complexity of the calculation of \( R_E \) from equation (8.10) depends linear on the amount of data \( m \) of the considered training set. Therefore, the calculation of the final rating \( R \) (equation (8.18)) relies on the total amount of data which can be seen as a conjunction of number of training sets \( p \) and (average) amount of values \( m \) (equation (8.19)). This linear dependency allows the processing even of large datasets.

\[
R_E \in \mathcal{O}(m) \quad R \in \mathcal{O}(p \cdot m) \quad (8.19)
\]

### 8.3.5 Fitness Function and Selection

Based on the final rating value \( R \) (equation (8.18)), the fitness function \( F : \mathbb{R} \rightarrow [0, 1] \) of an individual \( c \) can be set up. In a first step, an intermediate fitness value \( F' \) is determined depending on two boundary values. In equation (8.20) \( \hat{R}^{\min} \) and \( \hat{R}^{\max} \) denote minimum and maximum threshold values for the rating value \( R(c) \) of the current chromosome. These thresholds can either be set fix (e.g. \( \hat{R}^{\max} = 0 \) and \( \hat{R}^{\min} = -10^{10} \) depending on the expected range) or dynamically based on lowest and highest rating values of individuals inside of the current population. To keep the immediate fitness value inside of the boundaries of \([0, 1]\) they have to be limited in case of static threshold values (see appendix A.1.2 for the corresponding notation).

\[
F'(c) = \langle \frac{R(c) - \hat{R}^{\min}}{\hat{R}^{\max} - \hat{R}^{\min}} \rangle \quad (8.20)
\]
In case of dynamic thresholds, which are used here, the best individual (with \( R(c) = \hat{R}^{\text{max}} \)) receives an intermediate fitness of 1 whereas the function of the worst individual results to 0. Therefore, an offset value \( F^{\text{bas}} \) is used as basic fitness for all \(|P|\) individuals of that population to ensure that even the worst individual has a certain chance to survive. Equation (8.21) shows the final calculation of the fitness of a chromosome \( c \) based on the intermediate fitness as given before.

\[
F(c) = \frac{F'(c) + F^{\text{bas}}}{1 + F^{\text{bas}}} \tag{8.21}
\]

The next step is to decide, which individuals will be taken for the next generation. The chance of an individual \( c \) to survive depends on the ratio of individual to population fitness as already shown in equation (8.7). The algorithm now generates a random value \( r \in [0, 1] \) which selects an individual to be used in the next generation as given in equation (8.8). Since the new generation should contain the same amount of individuals, this step is repeated \(|P|\) times. Depending on its probability value \( p(c) \) and the result of the random values an individual can be used several times for the new generation or be removed.

### 8.3.6 Genetic Operations

Now the survivors (children) have to be adapted. As described in section 8.3.1, there exist two general ways of manipulating the chromosomes: Mutation and crossover. As described by [Spears1993], both operators have different characteristics, which can be summed by the question for exploitation or exploration. The present approach uses different mutation types to adapt the weights (genes) of the individuals. Crossover is not considered due to several aspects. The usage of crossover requires a diverse population, which would result in a large number of individuals to achieve a minimum of different weight values per gene. Since the approach uses floating point genes the gene pool is limited in every case, if only crossover is used. It is also not clear if there exist genes which need to be preserved or which good individuals have in common – even the existence of one suitable set of weights for the given training sets is not assured. Therefore, the exploration of the search space is most important. Spears postulates it as follows: “If optimality is sought, crossover may be deleterious. If the maximization of accumulated payoff is sought, mutation may be insufficient” [Spears1993, page 231]. In fact, three different mutation types with the given probabilities are used for weight adaption for each individual in each evolution step:

**Random weight offset:** Updates a gene (weight) \( w_i(s) \) of a chromosome \( c \) from step \( s \) with a random offset \( w_i^{\text{off}} \) according to equation (8.22). All offset values are distributed uniformly over a small range (e.g. 10% of the total weight range) and determined randomly for each gene\(^5\). The used probability, that a weight of an individual is changed, is determined by a parameter \( p^{\text{off}} \). For each gene, the mutation probability \( p^{\text{off}}/|c| \) depends on the total number of genes \(|c|\) inside of the chromosome \( c \), so it might e.g. happen that half of the individual’s weights are changed or none.

\(^5\)To be precise, the range of the offset value \( w_i^{\text{off}} \) is reduced if there exists the chance that the final weight \( w_i(s + 1) \) would exceed its limits. In this case, the upper or lower limit of the offset range is adapted and the offset value is determined randomly out of this new range.
This mutation mechanism is responsible for a local search in a close range of the individual. An example showing this mutation type is given in figure 8.7 which makes use of the weight and offset ranges as they are applied in the present case. The weights are $w_i \in [-1, 1]$ whereas the offsets are randomly set to a range of $w_i^{\text{off}} \in [-0.1, 0.1]$.

$$w_i(s + 1) = w_i(s) + w_i^{\text{off}}$$ (8.22)

Random weight: This mutation type updates a weight with a completely new random value $w_i^{\text{rand}}$ with probability $p_i^{\text{rand}}$. Again, the probability for each weight to be changed is $p_i^{\text{rand}}/|c|$. As shown in equation (8.23), the new weight $w_i(s + 1)$ does not rely on the previous weight. Related to the changed gene this seems to be a large evolutionary step, but compared to a complete individual with 40 or more genes, this update remains small. Nevertheless, this mutation type is important to escape from local minima and to allow a wider evolutionary spread. Figure 8.8 gives an example for this random mutation. Since the random weight has to cover the complete weight range it is also set to $w_i^{\text{rand}} \in [1, 1]$.

$$w_i(s + 1) = w_i^{\text{rand}}$$ (8.23)

Random multiplication: This mutation type, as shown in equation (8.24), updates all weights$^6$ of a chromosome $c$ with the same random multiplication factor $\kappa^{\text{mul}} \in [0.9, 1.1]$. Since this operation affects the complete individual, the probability $p_i^{\text{mul}}$ is given as the chance that one individual is updated inside of the complete population.

$^6$If a weight exceeds its range after this operation, it is limited by the two boundaries ($-1$ and $1$ in the present case).
Therefore, the chance of one individual to be updated is $p_m^{mul}/|P|$. This type of mutation becomes helpful in the first generation steps to lower or raise the complete evaluation function $E(t)$ (compare figure 8.6) without changing the overall structure of it\textsuperscript{7}. An example for this random multiplication can be found in figure 8.9.

$$w_i(s + 1) = w_i(s) \cdot \kappa^{mul} \forall w_i \in c$$

(8.24)

\begin{tabular}{c|c|c|c|c|c|c}

<table>
<thead>
<tr>
<th>$c'(s)$</th>
<th>0.34</th>
<th>0.64</th>
<th>1.00</th>
<th>-0.47</th>
<th>0.03</th>
<th>...</th>
<th>0.73</th>
<th>-0.03</th>
<th>-0.32</th>
</tr>
</thead>
</table>

random multiplication

<table>
<thead>
<tr>
<th>$c(s + 1)$</th>
<th>0.32</th>
<th>0.61</th>
<th>0.95</th>
<th>-0.45</th>
<th>0.03</th>
<th>...</th>
<th>0.70</th>
<th>-0.03</th>
<th>-0.30</th>
</tr>
</thead>
</table>

\textbf{Figure 8.9:} Example for an individual which is mutated completely by adapting all weights with a random multiplication factor (in this case 0.95).

Since the selection of genes or individuals is randomly, it is possible that some individuals remain unchanged whereas others mutate at several genes. The used probabilities for the three mutation types are given in section 8.4.2 containing experimental results of the optimization process. Nevertheless, all genes have to remain inside the search space. If one weight is near to one limit inside of its range, the possible weight offset has to be reduced in direction of the closer limit. In the same sense, the random multiplication has to be adapted for single genes or for the complete individual.

\subsection*{8.3.7 Elitism}

As shown before, there exist some reasons to use mutation only and neglect crossover operations. But, even the selection of an optimal mutation rate is not easy since it has influence on genetic drift or the withdrawal of good solutions. In every case, the population should get the chance to expand in all directions but be able to keep the best individual inside. A common approach for handling the problem of withdrawal of good results are elitism methods which ensure the surviving of the $i \in \mathbb{N}$ best individuals of the population.

\textbf{Figure 8.10:} Visualization of the elitism method: There exists a certain chance that an individual is replaced by the best one (elite individual) after the mutation step.

\textsuperscript{7}In the strict sense this is only the case if all weights have the same leading sign.
The existing optimization system uses a special kind of elitism function as illustrated in figure 8.10. Since weight determination – as it is used here – needs only the best set of weights, the best individual that has been found so far is stored (outside of the population). After the mutation step there now exists a certain probability $p_e \in \mathbb{R}$ for each existing individual, that it will be replaced by the injected elite individual. This chance of embedding decreases over time if no better individual has been found to enlarge the evolution space and to reduce the effect of local minima.

### 8.3.8 Breakpoint of the Optimization Process

The shown steps of weight evaluation, rating of each individual, selection via fitness function, mutation, and elite injection have to be repeated several times. The remaining question is now, how many steps have to be executed. In general, the ideal breakpoint of learning procedures is not easy to determine due to problems like over-fitting or local minima. Common approaches compare the benefit of the optimization steps or use certain thresholds to stop the procedure.

In the present setup two minimum requirements exist: There must be no false positive (no false alarm in situations of acceptable safety) in the used training sets and no false negatives (all potential drop-offs must be detected). The algorithm reaches this breakpoint $R_{bp}$, if no evaluation rating $R_E$ (equation (8.10)) of the best individual gained a malus $M_{E}^{\text{unw}}$ or $M_{E}^{\text{req}}$. In colloquial speech, this individual generates all required and no unwanted values. This happens at the latest if $\sum (R_{E}^2) < 10^8$ in equation (8.18), depending on the chosen malus. The minimum breakpoint is given in equation (8.25) and is not reached for sure until all malus values are thrown out of the system.

$$R_{bp} > -\sqrt{\frac{1}{p} \cdot (10^8)^2}$$  \hspace{1cm} (8.25)

So far, the algorithm is stopped manually if the results are sufficient and seem to be a good solution for the given training sets. Of course, it might become necessary to get additional training data and continue or restart the learning process with the new data sets in addition, if the error rate during operation is too high. The next section will present some experimental results concerning the error rates and the optimization process itself.

### 8.4 Experimental Results

The presented approach seems to be a suitable way to solve the given problem of online safety analysis and risk prediction. But, the proof, that this method really works is still missing. The following sections will present some experimental results which are performed in the simulated environment since this large number of tests cannot be executed on the real robot. This simulation uses a three-dimensional representation of the robot and its surrounding and is able to simulate chamber leakages based on the surface structure, chamber pressure variations including the air flow and the complete control network of CROMSCI with its closed-loop controllers and high deliberative functions as shown in appendix A.7.2.
8.4.1 Experimental Setup

In the first experiments, the simulated robot is driven downwards a wall on a rough surface as illustrated in figure 8.11. Here, the system either drives down without larger disturbances – e.g. on a rough but non-critical surface patch as shown in figure 8.12c – or it faces different defects in form of deep cracks causing a drop-off in the worst case as depicted in figure 8.12a and figure 8.12b. During these test runs, the behavior data and corresponding adhesion scores are logged to be used as training data according to the descriptions in section 8.2.2. Afterwards, these training examples are used to find suitable weights for the behavior evaluation function. But, before the result of the risk prediction is analyzed, first the execution of the optimization process itself will be examined.

![Visualization of the simulated environment](image)

**Figure 8.11:** Visualization of the simulated environment: This initial situation is the starting point for experimental runs to collect training data.

![Different simulated surfaces](image)

**Figure 8.12:** Different simulated surfaces: A straight horizontal crack (a), a more complex crack structure (b) and a basic surface (c) whose characteristics including roughness and grooves is also used for the other surface patches.
8.4.2 Optimization Process

The main questions of the complete optimization process are, if an individual – a set of weights – exists which can be used for the given task of risk prediction, if it can be found by the given approach, and how long it takes to find one. Therefore, a set of ten training examples is used to determine the needed weights for the evaluation function. In all ten examples, the robot was driven down a wall, two times nearly without disturbances on surface patch as shown in figure 8.12c and eight times with larger surface irregularities like given in figure 8.12a, which cause a drop-off in seven cases. In all these experiments, the weight range is $w_j \in [-1,1]$ and the random value for the offset mutation operation lies between $-0.1$ to $+0.1$.

![Figure 8.13: Optimization process of different test runs with $|P| = 100$ individuals per population. The used probabilities are $p_{\text{off}} = 0.75$ and $p_{\text{rand}} = 0.5$, the chance of an elite injection is $p_e = 0.1$. The thick lines represent the average of the ten ($p = 10$) test runs with ($p_{\text{mul}} = 0.1$) and without multiplication mutation ($p_{\text{mul}} = 0.0$).](image)

Figure 8.13 shows the decreasing evaluation rating $R$ of the best individuals over time for populations with 100 individuals and parameters $p_{\text{off}} = 0.75$, $p_{\text{rand}} = 0.5$ and $p_e = 0.1$. The breakpoint $R_{10}^{bp} \approx -3.162 \cdot 10^7$ is reached in most cases, although the rating values after 2200 generations still differ up to a dimension of 1000 between best and worst elite individual. It has to be mentioned, that the black lines indicate experiments with multiplication mutation $p_{\text{mul}} = 0.1$ as well as without ($p_{\text{mul}} = 0.0$). In average, both setups do not differ too much: The thick lines show the average values of the results with

---

8The breakpoint $R_{10}^{bp}$ depends on the number of training sets $p$ and is set to $R_{10}^{bp} = -\sqrt{\frac{1}{10} \cdot (10^8)^2}$ according to equation (8.25) in this case.
(green) and without the multiplication mutation (red) and reach the breakpoint after the considered 2200 cycles. It can be seen that the populations using the multiplication mutation deliver better results at the beginning of the optimization process, but the populations with $p_{m}^{mul} = 0$ are able to close the gap over the remaining time.

To compare different parameter setups a multitude of optimization runs has been executed. Table 8.2 and figure 8.14 give an overview on the optimization characteristics depending on population size, mutation rates and further settings. The complete plots of the experiments can be found in appendix section A.9. In figure 8.14 the different mutation types *random weight* and *random weight offset* deliver very similar results, even the amount of mutability does not have a large influence, if the sum of both lies within the range of about $(p_{m}^{off} + p_{m}^{rand}) \in [1, 2]$. Finally – as also shown in table 8.2 – the average values of the most test runs are very similar after the processed 2200 evolution steps as given by the black, green and blue lines in figure 8.14.

![Figure 8.14](image)

**Figure 8.14:** Comparison of the average ratings of the experiments based on the same ten training sets using different parameters like mutation rates, population size or elite chance.

The chance of elite injection and the population size have a stronger influence on the optimization speed. The complexity of one evolution step does not only rely on the number of training examples and the amount of data as given in equation (8.19) but also on the number of individuals, since the rating $R$ has to be determined for each individual to calculate its fitness value (compare section 8.3.5). Therefore, the calculation time per evolution cycle increases linearly with the amount of individuals. The dashed yellow graph represents experiments with 1000 individuals which perform ten times slower than the small populations. The two red graphs in figure 8.14 show populations with 10000 individuals which perform only one-hundredth of the evolution steps in the same time compared to the populations with 100 individuals. Their optimization process is much
### Table 8.2: Setup of the different experiments and results of the evaluation rating value $R$ after the given number of evolution steps based on 10-20 evolution trials each.

| $|P|$ | $P_{off}^{m}$ | $p_{rand}^{m}$ | $p_{mul}^{m}$ | $p_e$ | #s    | avg     | min (worst) | max (best) |
|-----|---------------|----------------|---------------|-------|-------|---------|------------|------------|
| 100 | 0.75          | 0.50           | 0.10          | 0.100 | 2200  | -1.607·10^4 | -1.023·10^5 | -8.530·10^4 |
| 100 | 0.75          | 0.50           | 0.00          | 0.100 | 2200  | -1.461·10^7 | -7.928·10^7 | -2.294·10^5 |
| 100 | 1.00          | 0.00           | 0.00          | 0.100 | 2200  | -1.662·10^7 | -7.868·10^7 | -2.495·10^4 |
| 100 | 0.00          | 1.00           | 0.00          | 0.100 | 2200  | -1.032·10^7 | -2.098·10^7 | -2.924·10^5 |
| 100 | 0.75          | 0.25           | 0.10          | 0.100 | 2200  | -3.207·10^7 | -2.261·10^8 | -8.485·10^4 |
| 100 | 1.50          | 0.50           | 0.00          | 0.100 | 2200  | -1.756·10^7 | -5.301·10^7 | -5.384·10^5 |
| 100 | 0.75          | 0.50           | 0.10          | 0.001 | 2200  | -1.031·10^8 | -2.230·10^8 | -1.755·10^7 |
| 1000 | 0.75         | 0.50           | 0.10          | 0.010 | 220    | -9.409·10^7 | -3.057·10^8 | -1.177·10^7 |
| 10000 | 0.75        | 0.50           | 0.10         | 0.100 | 22    | -1.326·10^8 | -2.491·10^8 | -4.333·10^7 |
| 10000 | 0.75        | 0.50           | 0.10         | 0.001 | 22    | -2.224·10^9 | -6.701·10^9 | -2.457·10^8 |
| 100  | 0.50          | 0.50           | 0.00          | 0.100 | $10^9$ | -6.128·10^9 | -1.605·10^9 | -1.555·10^9 |

slower, in most cases the best individual of large populations did not reach the breakpoint after the chosen period of time, as it can be seen in figure A.24 at the appendix. Beside the population size also a small or zero elite chance has negative influence on the optimization process as shown by the solid yellow graph in figure 8.14 which increases not as fast as the other graphs.

![Figure 8.15](image_url)

**Figure 8.15:** Long optimization runs using the same ten training sets with more than 100,000 steps which took about 19 hours each. The used settings are $|P| = 100$, $P_{off}^{m} = 0.5$, $p_{rand}^{m} = 0.5$, $p_{mul}^{m} = 0.0$ and $p_e = 0.1$.

For completeness, five long optimization runs with a duration of more than 19 hours have been executed as given in figure 8.15. These experiments show that a nearly ideal set of weights with an average evaluation rating $R$ of about -6,000 can be found, if the process
has enough time for optimization\textsuperscript{9}. For the application, this means that the trained weights lead to an evaluation function with nearly the optimal characteristics to avoid false positives but also to signal an upcoming robot drop-off in the desired time range.

According to the short experiments with an optimization time of about 1500 seconds, those experiments deliver the best average value and the lowest minimum rating (figure 8.14), which use only random weight mutation $p_{m}^{\text{rand}} = 1.0$ as genetic operation, whereas the experiments with random offset weight mutation $p_{m}^{\text{off}} = 1.0$ got the best maximum rating as given in table 8.2. This seems to be a problem of too few statistical data since 10-20 evolution trials per setup might not be sufficient for further analysis. But, the data show the benefit of small populations and the application of the elitism method.

The presented optimization process using a genetic algorithm has also been tested against pure random search [Zabinsky2011]. This approach is able to find the same weights as the presented method and therefore might be suitable, too. But, compared to the genetic optimization pure random search has a much lower performance. Of course, it is possible, that the first set of random weights delivers the desired characteristics\textsuperscript{10}, otherwise is the chance relatively low compared to the large search space. In fact, pure random search corresponds to a special setup of the genetic algorithm with only one individual ($|P| = 1$), a guaranteed random mutation of each weight ($p_{m}^{\text{off}} = 0.0$, $p_{m}^{\text{rand}} \approx 100000.0$, $p_{m}^{\text{mult}} = 0.0$) and no elitism ($p_{e} = 0.0$). Figure 8.16 illustrates the optimization process of pure random search with one million iterations per test run. It can be seen that it converges less fast than the previous approach. Even the best set of weights of ten million trials is still factor 100 above the optimization break point $R_{10}^{bp} \approx -3.162 \cdot 10^7$. This result proves the efficiency of the developed optimization method.

\textbf{Figure 8.16:} Optimization process using pure random search with the same training sets.

\textsuperscript{9}For comparison: The breakpoint in this experiment was at about -31 620 000.

\textsuperscript{10}Of course, this counts also for the individuals of the initial population.
8.4.3 Risk Prediction

To evaluate the training results additional tests are examined in which the robot is confronted to similar situations as in the training phase. The used set of weights is described in appendix A.8. Figure 8.17 depicts the reaction times $t_{\text{react}}$ of 64 test runs in which the robot is driven down the wall facing a deep crack (figure 8.12a). The bars in this figure represent the number of experiments in which the reaction time lies between the lower and upper discretization limits in steps of 100 ms. Again, the evaluation value $E$ should signal a drop-off early enough to have enough time for counteractive measures. Figure 8.17 shows one experiment with a reaction time below 400 ms and one with $t_{\text{react}} \in (3700, 3800]$. Although the evaluation signaled an upcoming drop-off in both cases, these values are not within the acceptable range since they are either too late or too early (tending to a false-positive). In total, 19 values of the 64 (30\%) lie inside of the trained range $[\Delta t, 2\Delta t]$ which seems not to be very good. Considering an acceptable range of $[0.5\Delta t, 3\Delta t]$, which is suitable for a prediction and necessary counteractive measures, nearly all values (97\%) are suitable. This is impressive to such a degree as the driven trajectory and local disturbances differ in all these cases.

Figure 8.17: Experiment 1: Reaction times of 64 test runs on the same deep crack as in the case of the training data while the robot drives down the wall.

In a second test run, the robot is faced with a similar situation as in the first case, but with a much more complex crack structure (figure 8.12b) which changes the behavior and the reaction of the adhesion system. Figure 8.18 shows the results of 49 experimental runs. One can see, that - compared to the previous case - more reaction times lie outside of the acceptable range and some predictions failed completely. In total, 34 reaction times are valid whereas 10 are invalid and 5 failed. This is worse compared to the previous case, but it has to be considered that this crack structure has not been trained, so this experiment shows that a transfer of training results on new situations is possible, but of course not optimal. Therefore, the used training examples are extended by three additional training sets in which the robot drives on this new crack structure. Again, the weights are acquired via the genetic algorithm to optimize the rating function.
After training, the robot tries to overcome the described cracks again. Figure 8.19 shows the results of the online analysis, which are significantly better compared to the previous experiment. Only in one case out of 29 the reaction time lies outside of the acceptable range. This example shows the necessity, but also the benefit of a certain amount of training sets.

Figure 8.18: Experiment 2a: The robot is driven down the wall 49 times, but at a much more complex crack structure which has not been trained explicitly. The star (⋆) indicates experiments in which the reaction times were above four seconds.

Figure 8.19: Experiment 2b: The robot is driven down the wall at the complex crack structure, but with three additional training examples at this surface patch.

Beside the correct detection, also the avoidance of false-positives is important. Therefore, the reaction on rough terrain with defects has been tested, which do not lead necessarily to a drop-off. In a third experiment, the robot drives over a rough surface patch as it has been considered in two of the ten training examples and shown in figure 8.12c. Figure 8.20 presents the results in form of a comparison of maximum adhesion score $S_A$ and maximum evaluation value $E$ during each of the 36 test runs. In total, 32 true negatives (green dots)
could be achieved in which no drop-off has been predicted correctly. In two cases the disturbances based on the rough structure and grooves nearly resulted in a drop-off of the robot ($S_A > S_{A^{\text{haz}}}$) which has been detected as hazard correctly (yellow dots). Nevertheless, the prediction generated also two false positives in which the evaluation value $E$ reaches 1 (red dots), but without that the robot is in too large danger.

![Figure 8.20: Experiment 3: False positives (red), true negatives (green) and true positives (yellow) while driving on a rough surface.](image)

Based on the number of true positives ($\#tp$), false negatives ($\#fn$), true negatives ($\#tn$) and false positives ($\#fp$), characteristic rates of the performance of the prediction can be determined. Table 8.3 sums up the experimental results by statistical rates like sensitivity $T_P$, miss rate $F_n$, specificity $T_n$ and the rate of false alarms $F_p$. $T_c$ describes the accuracy of the prediction which can be calculated, as well as the other rates, as given in appendix section A.1.3. Based on experiments 1, 2b and 3, a high accuracy in case of the trained situations can be attested.

<table>
<thead>
<tr>
<th>experiment</th>
<th>classification</th>
<th>rates</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$#tp$</td>
<td>$#fn$</td>
<td>$#tn$</td>
<td>$#fp$</td>
<td>$T_P$</td>
<td>$F_n$</td>
<td>$T_n$</td>
</tr>
<tr>
<td>1</td>
<td>62</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0.969</td>
<td>0.031</td>
<td>-</td>
</tr>
<tr>
<td>2a</td>
<td>34</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0.694</td>
<td>0.306</td>
<td>-</td>
</tr>
<tr>
<td>2b</td>
<td>28</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.966</td>
<td>0.034</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>32</td>
<td>2</td>
<td>1.000</td>
<td>0.000</td>
<td>0.941</td>
</tr>
<tr>
<td>total$^a$</td>
<td>126</td>
<td>18</td>
<td>32</td>
<td>2</td>
<td>0.875</td>
<td>0.125</td>
<td>0.941</td>
</tr>
<tr>
<td>trained$^b$</td>
<td>92</td>
<td>3</td>
<td>32</td>
<td>2</td>
<td>0.968</td>
<td>0.032</td>
<td>0.941</td>
</tr>
</tbody>
</table>

$^a$ The total values are calculated based on all experiments.

$^b$ The trained values are based on experiments 1, 2b and 3 and sum up the results of trained situations.
8.4.4 Limitations

One main aspect still remains unconsidered: Although the previous examples have shown that it is possible to find suitable weights, there is no guarantee that this works for all cases. Further experiments have shown, that the behavioral situation is completely different if the robot e.g. drives upwards. Figure 8.21 shows the result of a couple of test runs in which the robot drives up a wall with the already trained weights. In 14 of 18 cases the forthcoming detection of an upcoming hazard failed, which makes it necessary to add also these cases to the pool of training sets.

Unfortunately, it is not possible to determine a set of weights fitting to all situations and training examples. In the present case of driving up a wall and driving down, it is hard or impossible to find weights which can be applied in both situations as depicted in figure 8.22. In this case, the genetic algorithm tries to find a set of weights which fits 15 different training sets. In contrast to the previous cases, these training examples combine experiments in which the robot is driven downwards and upwards. According to equation (8.25) the breakpoint is \( R_{15}^{bp} = -2.582 \cdot 10^7 \). Even after 50 000 evolution steps (more than 16 hours) no rating value \( R \) has reached this breakpoint. In absolute values, the average lies at \(-554\,977\,348\) while the best rating is \(-245\,815\,576\), which is still factor ten away from the breakpoint.

This aspect is mainly based on the different reactions of the adhesion behaviors, especially of the behaviors which are responsible for force control. To balance out the general robot tilt as introduced in section 7.5.1 the upper chambers need a higher negative pressure than the lower ones. If the robot drives down the wall and reaches a large defect those chambers lose their leak tightness and are exposed to ambient pressure first, which do not contribute much to robot adhesion since their negative pressure is low. In contrast to that, the chamber with the highest amount of adhesion force is exposed to a deep crack first if the robot drives up a wall. The impact on the robot system is higher due to a larger loss of adhesion force and an increased tilt effect.
8.4. Experimental Results

![Graph showing evaluation rating over generation](image)

**Figure 8.22:** Long optimization runs trying to optimize the rating function based on 15 training sets ($p = 15$). The used settings are $|P| = 100$, $p^\text{off} = 0.5$, $p^\text{rand} = 0.5$, $p^\text{mul} = 0.0$ and $p_e = 0.1$.

8.4.5 Situational Risk Prediction

Based on the previous results, situation dependent weights are needed, which have to be trained for these different cases. Up to now, the complete risk prediction system considers three different situations which have to be trained independently: Driving down the wall, driving up and driving horizontal. In general, robot movement is a combination of a motion in horizontal and vertical plane. If the robot e.g. drives upwards and left, this velocity vector can be split up into a horizontal and a vertical proportion related to the vertical surface. Therefore, the prediction method needs to be sensitive for these two situations and has to identify potential hazards for upward driving as well as for horizontal motion. First of all, the direction of robot motion related to the environment $[E] \varphi_{R|v}$ has to be determined (equation (8.26)).

$$
[E] \varphi_{R|v} = \left([R] \alpha_{v_R} + [E] \varphi_R \right) \mod (2 \cdot \pi) \tag{8.26}
$$

with $[R] \alpha_{v_R} = \text{atan2} \left([R]v_{R|y}, [R]v_{R|x} \right)$

Here, $[R]v_{R|x}$ and $[R]v_{R|y}$ are the velocity of the robot in its own x- and y-coordinates as depicted in figure 8.23, $[R] \alpha_{v_R}$ is the direction of robot velocity and $[E] \varphi_R$ denotes the robot orientation on the vertical wall (related to the environment). Finally, the angle is transformed into the range of $[0, 2\pi)$ via a modulo operation according to equation (A.4). Although this angle of motion already denotes the direction of motion, it does not say anything about the velocity. Therefore, the specific robot motion $[E]v_{R}$ in the environmental frame is calculated as given in equations (8.27) and (8.28).
These two velocity values from equations (8.27) and (8.28) are then used to distinguish between five motions: Driving up, driving down, driving sidewards, driving up and sidewards, or driving down and sidewards. The situation dependent evaluation value $E_S$ is then either one single evaluation value or a combination of two values $E_i$ (see equation (8.2)) which have to be selected according to equation (8.29) with a velocity threshold $\hat{v}_{RE}$ to ignore too slow motions. In the case of a stopped robot (else-case in equation (8.29)), the maximum of all evaluation values is used to receive a suitable value without training a fourth set of weights for a non-moving robot.

$$
E_S = \begin{cases} 
E_{\text{up}}, & \text{if } (|E_{VRx}| > \hat{v}_{RE}) \land (|E_{VRy}| < \hat{v}_{RE}) \\
E_{\text{down}}, & \text{if } (|E_{VRx}| < -\hat{v}_{RE}) \land (|E_{VRy}| < \hat{v}_{RE}) \\
E_{\text{side}}, & \text{if } (|E_{VRx}| < \hat{v}_{RE}) \land (|E_{VRy}| > \hat{v}_{RE}) \\
\max(E_{\text{up}}, E_{\text{side}}), & \text{if } (E_{VRx} > \hat{v}_{RE}) \land (E_{VRy} > \hat{v}_{RE}) \\
\max(E_{\text{down}}, E_{\text{side}}), & \text{if } (E_{VRx} < -\hat{v}_{RE}) \land (E_{VRy} > \hat{v}_{RE}) \\
\max(E_{\text{up}}, E_{\text{down}}, E_{\text{side}}), & \text{else}
\end{cases}
$$

To include similar cases like driving left and driving right without learning them explicitly an additional mechanism is implemented which mirrors the weights which are applied to behavior values which have a horizontal twin. In a first step, all behavior values which are related to a specific chamber like chamber control behavior or chamber leakage estimation are transformed into a global view relative to the environment depending on the robot orientation. As given in the example in figure 8.24a, the values of the highest chamber – here chamber $C_5$ – are mapped to top values (figure 8.24b) whereas chamber $C_2$ is the bottom chamber. This step is used both during collecting the training data as well as later on if the behavior values are evaluated, and allows it to train situations independent from the real robot orientation since it does not have a preferred direction of motion.

![Figure 8.23: Determination of the direction of motion related to the environment based on velocity vector $[R]v_R$ and robot orientation $[E]\varphi_R$.](image-url)
The second step is to make use of the symmetrical setup of the robot since there is no distinction needed between left and right (figure 8.24c). In fact, it does not make any difference if the robot drives left facing a crack or a rough terrain or if it drives right facing the same situation. Therefore, the evaluation function of each of the three directions up, down, and side depends not only on the original learned weights, but also on a set of weights in which the horizontal elements – which refer on behaviors related to chambers on the left and on the right side – are substituted by their opposite 'twin' as depicted exemplarily in figure 8.25. Finally, the evaluation value $E_i = \max(E_i(\vec{w}), E_i(\vec{w}_{mirror}))$ is the maximum of both single evaluations based on the same input data, but on different sets of weights. This allows it to train driving left and use the results also for driving right or vice versa. The resulting sets of weights can be found in appendix A.8.

Figure 8.25: Example of mirrored weights, which are created based on the resulting weights of the optimization process, and appropriate meta values.
Beside climbing on vertical or overhanging walls the robot is also able to drive on the ceiling. In this case, a fourth – but neither implemented nor tested\textsuperscript{11} – situation with its own training sets has to be considered. Although an upside-down driving has not been performed so far, it is expected, that the system will perform better than in the vertical case. This statement is based on the fact, that the drive system has not to overcome gravity and that there exists no (or only a neglectable) robot tilt caused by its mass distribution.

8.4.6 Discussion

This chapter presented a method of risk prediction for safe navigation of a wall-climbing robot, which is based on meta data of a behavior-based control network [Schmidt2012b]. This prediction calculates a risk value which points out if the robot is going to fail within the next seconds or not, and makes use of a weighted sum of the used behavior values. To determine suitable sets of weights some training sets are used to optimize an evaluation rating value. In practice, the evaluation system has to be trained once and can be applied to similar situations and setups. Nevertheless, a situation dependent prediction is needed which differs between the three main motion directions up, down and sidewards. The results of the complete system regarding the detection and false alarm rates are very promising and allow a good prediction of upcoming hazards. In terms of safety, this analysis and prediction is essential as demanded in requirement 15 (section 5.2.3) so that the robot is able to identify hazardous situations soon enough to initiate counteractive measures. Of course, this detection is only the first part. Possible measures based on this evaluation – to avoid these unwanted events and to minimize the probability of a robot drop-off – will be presented in the next chapter.

\textsuperscript{11}Experiments on the ceiling have not been performed because of an unavailable training ground and a larger effort in the security system, which has to protect the robot against a drop-off via several ropes.
9. Risk Handling Measures and Strategies

The previous chapter presented a method for risk prediction, which analyzes the present state of the robot control system to identify upcoming dangerous situations caused by the current terrain structures. But a detection is not very helpful if the corresponding reaction is missing. This chapter will introduce counteractive measures, which will improve robot safety in a way that the system is able to avoid detected risks completely. This includes those hazards as identified via the genetic algorithm approach as well as hazards caused by macro structures which can be detected more easily and beforehand. The first section 9.1 will introduce the detection of obvious risks caused by larger obstacles and holes and corresponding methods to avoid them. Afterwards, section 9.2 presents counteractive measures related to the risk prediction from the previous chapter which try to avoid or reduce the effects of these hardly foreseeable events. Experimental results with the real prototype CROMSCI related to risk prediction and safety measures are given in section 9.3, the benefit for robot navigation safety will be summed up in section 9.4.

9.1 Avoidance of Obvious Risks

As presented in chapter 3, the climbing robot is not only endangered by hardly predictable hazards (e.g. caused by unforeseen or (nearly) undetectable surface features) or by omnipresent effects like wheel slip. Similar to other mobile vehicles, climbing robots also have to deal with obvious risks caused by larger macro structures. But in contrast to them climbing systems are only confronted with static obstacles which can be classified in two major types: Positive obstacles like protruding objects or overlapping structures, and negative ones (holes) in form of single deepenings or structure edges. This distinction is caused by a requirement concerning the manipulator arm, because it is necessary to prevent a collision with positive obstacles, but it is obviously no problem if the arm reaches over holes. The influences of these external hazards can be described well as given in sections 3.4.2 and 3.4.3. Also detection and handling of these hazards is easier compared to others.
A common way to enhance robot navigation safety related to these macro objects are methods for evading obstacles automatically. Many of these systems use planar laser range sensors in two or three dimensions, obstacle detection algorithms and reactive measures to avoid them. In contrast to ground-based mobile robots [Philippen2003, Aboshosha2003, Schmidt2006] climbing robots have special requirements and limitations concerning the obstacle avoidance system:

- The number and type of environmental sensors which can be applied is limited due to very restricted payload and mounting space which reduces the perception capabilities tremendously. Laser range sensors by Sick AG\(^1\) e.g. are often used in robotics, but with a weight between one and four kilogram they are still too heavy for this application.

- Another interesting point arises from the omnidirectional movement of the climbing robot, which allows different kinds of evasion strategies. Nevertheless, this includes that the environmental detection needs to work 360° all around the robot if it is able to drive in any arbitrary direction.

- As in the case of other mobile robots, the obstacle avoidance system is safety critical since it prevents the robot chassis and its superstructural parts from damages. But, in the present case, these collisions or an unrecognized hole could lead to a damaged and inoperable robot which needs to be rescued manually. The system must allow a reliable avoidance of these environmental features.

- Odometry for localization of a wheel-driven climbing robot is less accurate than in the case of ground-based vehicles, which makes it more difficult to create detailed maps. Additionally, some of further localization approaches like scan matching [Borrmann2008] or GPS cannot be applied here as well as an inertional measurement system [Koch2005] which fails because of robot vibrations. Solutions might be visual odometry, a passiv dead reckoning wheel [Hillenbrand2004] or an external laser beacon. Nevertheless, the mounting space and weight of these systems has to be considered.

### 9.1.1 Detection of Obstacles

Considering weight restrictions, CROMSCI has been equipped only with a light-weighted laser range sensor by Hokuyo\(^2\) of type URG-04LX which is the main sensor for obstacle detection. As presented in section 2.1.5 it has a total angle range of 240° and is attached on top of the manipulator arm with a fixed tilting angle of \(\theta_H = 20°\) to be able to rotate it. Therefore, it can be moved in the same way as the sensor head. Beside the real integration also a simulated version of this scanner exists. Figure 9.1a shows a visualization of the scan lines and its range. The resulting scan points measured by the sensor are depicted in figure 9.1b with prominent scan points notifying holes (see marks 1 and 2) if the scan points lie beyond the ground line or obstacles (marked with 3) if the points are in front of it.

\(^1\)http://www.sick.com/

\(^2\)http://www.hokuyo-aut.jp/
Figure 9.1: Visualization of the scanner lines in the simulated environment (a) and the resulting scan points in the laser ranger plane (b).

Obstacle Detection

The obstacle detection as published in [Jung2010] is performed by finding special edges in sensor distance data by comparing the indexed distance points \( x_P \) in scanner coordinates with the reference distance \( x_{P}^{\text{ref}} \) between ground line and scanner line and an offset value \( x_P^{\text{offset}} \). This leads to points of interest (POI) regarding equation (9.1) which are given to a mapping algorithm to store the obstacle positions in a local memory whereas missing obstacles indicating free space can be used to delete phantom obstacles.

\[
POI(x_P) = \begin{cases} 
  x_P, & \text{if } x_P < x_{P}^{\text{ref}} - x_P^{\text{offset}} \text{ (positive obstacle)} \\
  x_{P}^{\text{ref}}, & \text{if } x_P > x_{P}^{\text{ref}} + x_P^{\text{offset}} \text{ (negative obstacle)} \\
  0, & \text{else (no obstacle, free space)}
\end{cases} \quad (9.1)
\]

Based on the fact that obstacles can only be detected in a certain scan plane, they have to be stored within a local map used as a short-term memory. This map has to cover at least the area between the robot and the current detection distance and has to be abstract for a higher robustness against odometry errors as mentioned before. In the present case, a circular map with a diameter of 6 m is generated as depicted in the screenshot from figure 9.2a. This map contains the detected obstacles like the previous ones from figure 9.1. Also the current scan line is visualized to show at which position inside of the map new environmental informations are gathered. The diameter of the map results from the far-sight of the laser range sensor. Objects lying outside of the map will be forgotten.

Local Mapping

For obstacle avoidance, the positions of the holes and objects are transforming into a sector view as given in figure 9.2b. Here, the surrounding environment is divided into 36 sectors with an opening angle of 10° each. Each sector value is the closest distance of a POI inside of the sector range related to the robot center. Additionally, the type of obstacle (negative or positive) is encoded for further processing. Due to the fact that the manipulator arm performs a different kind of motion and that it sticks out of the round
robot chassis, an additional sector map is required. In contrast to the previous round sector map from figure 9.2, these sectors are relative to the position of the tool center point (TCP). Furthermore, two kinds of sectors exist based on different motion types: Circular and rectangular. Figure 9.3 shows the sectors related to the manipulator arm with areas in which the motions are slowed (yellow), and those in which the motion in the considered direction is not allowed (red) if an obstacle lies within the sector. The size of the red areas for sideward and rotating motion (figures 9.3a and b) depend on the current arm width which varies since the two sledges are driven independently (compare section 2.1.4), whereas the sectors itself are set to fixed sizes.

The circular sectors in figure 9.3a are used both to slow down rotating motions of the robot or a sideward motion of the manipulator. In fact, it makes no difference for the manipulator arm if it performs a circular motion and hits an obstacle or if the robot chassis is turned. Figure 9.3b shows the sectors which influence robot sideward motion to
9.1. Avoidance of Obvious Risks

Protect not only the robot chassis but also the manipulator arm from damages. Finally, the frontal sector in figure 9.3c again affects both robot motion in the certain direction and the extension of the TCP.

Concept of Obstacle Avoidance

The overall structure of the obstacle avoidance system includes all these elements like obstacle detection, local mapping, localization via odometry, and iB2C safety behaviors. Figure 9.4 depicts these four groups of control and processing components. The sensor values from laser ranger, motor encoders of the manipulator, and steering and drive encoders (icons from left to right) are given to processing elements performing transformations, filtering, or further calculations – either for obstacle detection (green) or for localization (yellow). Based on the mapped objects, the sector distances are calculated (red) which are given to safety behaviors (blue) in terms of virtual sensor information to allow a reactive avoidance of these hazards.

Figure 9.4: Structure of the obstacle avoidance system reaching from hardware components (bottom, compare table A.3) via processing elements up to mapping and emergency behaviors on top.
9.1.2 Behaviors for Evasive Actions

For obstacle avoidance, a robot needs suitable counteractive measures to react on the detected hazards. Based on the collected map data, different safety behaviors are supplied with sector distance information (figure 9.4) to either slow the robot down to prevent a collision or to let it evade the obstacle by turning away. If the sector information indicates a positive or negative obstacle, the behaviors become active and affect other basic control behaviors which are responsible for robot movement. Again, the used behaviors follow the design of the iB2C architecture (section 2.2.1) in which meta signals are used for behavioral interaction and to signal the current behavior state.

Slow Down of Robot and Manipulator Motions

Two different reflexive behaviors have been developed for collision avoidance. The first one is an autonomous deceleration based on the detected obstacles within a certain range (e.g., yellow areas in figure 9.3) which enables the robot to meet requirement 11 (section 5.2.2). The deceleration ends up in a total stop of the motion in the observed direction when reaching a minimal distance as given by the red areas in figure 9.3. As described above, there is a special treatment of motions of the manipulator arm, because it only has to be considered while moving towards positive obstacles whereas it can reach over gaps and holes. In total, there exists several instances of the basic slow down (SD) behavior, which are responsible for different directions of motion to allow movements towards obstacle-free regions. Each behavior receives the minimal distance value $d_{SD}^{min} = \min(d_{Mi})$ of the map sectors $M_i$ in the specific movement directions. Based on this distance, the target rating $r_{SD}$ and the activity value $a_{SD}$ are calculated according to equations (9.2) and (9.3). Within the control network, the activity values of these behaviors are used to inhibit the corresponding motion behaviors. The behavior values therefore reach from 0 (no stop) to 1 (full stop) using distance thresholds $d_{SD}^{near}$ and $d_{SD}^{far}$ for a near and far limit with $d_{SD}^{near} < d_{SD}^{far}$. $\iota_{SD}$ is the internal activation of the slow down behavior based on its stimulation and inhibitions. Figure 9.5 illustrates the development of the target rating value.

\[
\begin{align*}
    r_{SD} &= \begin{cases} 
    0 & , \text{if } d_{SD}^{min} \geq d_{SD}^{far} \\
    1 & , \text{if } d_{SD}^{min} \leq d_{SD}^{near} \\
    1 - \sin \left( \frac{\pi}{2} \cdot \frac{d_{SD}^{min} - d_{SD}^{near}}{d_{SD}^{far} - d_{SD}^{near}} \right) & , \text{else}
    \end{cases} \\
    a_{SD} &= \iota_{SD} \cdot r_{SD} \tag{9.3}
\end{align*}
\]

Automatic Evasion of Obstacles

But an automatic slow down is not the only option to avoid obstacles as demanded in requirement 10. Additionally, an autonomous evasion (EV) behavior has been developed to turn the robot away from obstacles and towards a more safe direction. It is important to react early enough on obstacles to leave enough space for maneuvers. The evasion behavior – which exists only once inside of the control network – calculates a weighted
9.1. Avoidance of Obvious Risks

Figure 9.5: Development of the target rating (and implicitly also of the activity) of the slow down behavior depending on the minimal distance value $d_{SD}^{\text{min}}$.

difference between sectors on the left and on the right side of the current driving direction to get the safest way. To achieve this, two weights $w_{EVL}$ and $w_{EVr}$ for the left and right side of the robot are calculated pointing out the amount of obstacles on this side. The weight value for the left side is determined as shown in equation (9.4). It sums up values of all considered $n$ map sectors ranging from most-left sector (index 0) to frontal sector (index $n-1$) corresponding to figure 9.2b. Here, $\mathcal{M}_l$ denotes a set of $n$ map sectors left of the current direction of motion. $d_{M_i}$ is the closest distance value of sector $M_i$ and $d_{EV}^{\text{far}}$ is a far distance limit. The same calculation is performed for the weight $w_{EVr}$ of the right sectors by using $M_i \in \mathcal{M}_r$.

\[
\begin{align*}
\frac{w_{EVL}}{w_{EVr}} &= \left(1 - \left(\frac{d_{M_i}^{\text{min}} - d_{M_i}^{\text{near}}}{d_{EV}^{\text{far}} - d_{EV}^{\text{near}}}ight)^i\right) \cdot \left(1 - \left(\frac{d_{M_i}^{\text{min}}}{d_{EV}^{\text{far}}}\right)^i\right) \\
\text{with } \quad M_i &\in \mathcal{M}_l \quad \text{and} \quad d_{M_i}^{\text{min}} = \min\left(d_{M_0}, d_{M_1}, \ldots, d_{M_{n-1}}\right)
\end{align*}\]  

(9.4)

The sector positions have an influence on the weight: Map sectors in motion direction (front) have a higher weight than those at the side of the robot. Additionally, the two weights $w_{EVL}$ and $w_{EVr}$ are scaled (weakened) depending on the closest measured value $d_{M_i}^{\text{min}}$ on the specific side. The side with the lower weight (more free space) is used for avoiding. The target rating $r_{EV}$ depends on the sum of both weight values (equation (9.5)) whereas activity $a_{EV}$ is determined via the internal activation $\iota_{EV}$ and the maximum of both weights (equation (9.6)):

\[
\begin{align*}
r_{EV} &= \left(\frac{w_{EVL} + w_{EVr}}{w_{EVL}}\right)^i \quad (9.5) \\
a_{EV} &= \iota_{EV} \cdot \left(\max\left(w_{EVL}, w_{EVr}\right)\right)^i
\end{align*}\]  

(9.6)

Since this behavior must be able to influence the single basic motion behaviors independently, it generates additional outputs. These limited weight values are used to stimulate a turning into the free direction and lie in the range of zero to one. In the case of $w_{EVL} > w_{EVr}$ (the distance rating on the left is higher) a turning to the right is invoked – and vice-versa.
Integration into Behavior-Based Control Network

The overall structure of the behavior network is very important for the control process. For obstacle avoidance, the basic motion behaviors are influenced by stimulation or inhibition by the *evasion* or the *slow down* behaviors. Figure 9.6 shows, how the additional safety behaviors are embedded into the existing control network. The *evasion*, for example, activates the turning behaviors in the amount of its own activity. The behaviors for straight motion or turning are inhibited by the corresponding slow down behavior at the bottom layer of the figure. The blue fusion behaviors on top of each basic movement behavior are needed to merge control inputs from different higher motion behaviors, which have the possibility to execute this kind of motion.

![Diagram of behavior-based control network](image)

**Figure 9.6:** The interaction of the basic behaviors for robot motion control and the integrated evasion (top) and slow down behaviors (bottom).

### 9.1.3 Experimental Results

To show the concept of behavior interaction and the operability of these safety measures, a couple of experiments has been executed both in simulation and with the real robot. Figure 9.7 shows results of the *slow down forward* (SDF) behavior in terms of different behavior values, the robot trajectory and the local obstacle map of the real prototype CROMSCI driving on the ground. For each behavior, all three state values *activation* $i$ (red), *activity* $a$ (green) and *target rating* $r$ (blue) as introduced in section 2.2.1 are plotted. During this experiment the robot was simply set to drive forward with at speed, which can be noticed by the raised green graph of the activity $a_{DF}$ of the *drive forward* (DF) behavior at (A). The *slow down forward* behavior is activated ($i_{SDF} = 1.0$) as indicated by the red bar, but it is still not active ($a_{SDF} = 0.0$). The low target rating $r_{SDF} = 0.0$ (blue graph) points out that the slow down behavior is satisfied at the beginning of the experiment. While the robot drives forward starting at the given position (dotted robot circle in figure 9.7), the distance to the obstacles decreases and the *slow down forward* behavior becomes unsatisfied (starting at (B)) and inhibits the forward driving using its
9.1. Avoidance of Obvious Risks

The robot slows down (a speed reduction of 95% is reached at about (C)) until it performs a full stop (D) at the final position to prevent a collision. To get away from this situation the robot has to be driven backwards or turned away manually via joystick control.

![Graph and Diagram](image)

**Figure 9.7:** The drive forward behavior is inhibited by the activity \( a_{\text{SDF}} \) slow down drive forward behavior depending on the distance to frontal obstacles. Finally, the robot velocity is reduced until a full stop.

In the second experiment a more complex situation is given and some additional behaviors are involved. Although the automatic velocity reduction via the slow down behaviors is satisfying to prevent the robot from collisions, it reduces the handling of the robot because it does not allow the robot to continue driving. Figure 9.8 shows the situation of the robot driving into a dead end (bottom). Above, the already introduced behaviors drive forward and slow down drive forward are plotted. In between, the values of the basic motion behaviors turn left (\( \text{tl} \)) and turn right (\( \text{tr} \)) as well as of the frontal evasion (\( \text{ev} \)) behavior are given. Here, the interaction of the behaviors can be observed, as soon as the robot detects the first obstacles: The activity \( a_{\text{EV}} \) of the evasion increases and stimulates the turn left behavior (A). As presented in previous section 9.1.2, this turning depends on the distances of obstacles on the left and right side of the robot. Additionally, the slow down drive forward behavior becomes active (B) because of the decreasing obstacle distance and slows down the forward driving as in the previous experiment. During the time from 22 s to 27 s (D) the robot nearly stays at the same position and turns to the
left $a_{TL} = 1.0$ until the way in driving direction is free again. For a short period of time between 18 s and 20 s (C), the robot was undecided which turning direction is the best which can be seen by the two peaks of $a_{TR}$. Due to some boundary conditions, which have been developed to avoid exactly these situations, the robot was able to manage this problem and finally turned left. At the end of the experiment (E) the robot leaves the dead end and is able to continue driving forward.

Figure 9.8: Experiment in which the robot drives into a dead end. The robot is slowed down and the evasion behavior automatically triggers a turning of the robot to the left.
9.2 Counteractive Measures

Beside the obvious risks, also the hardly predictable ones have to be handled. The difference between both is that macro structures can be detected beforehand in contrast to the surface related features. Therefore, other counteractive measures are needed in the second case. The next section will introduce these measures triggered by the estimated risk value, as determined in the previous chapter.

9.2 Counteractive Measures

By using the different sets of weights, which have been determined depending on three different driving situations as described in section 8.4.5, the robot system is able to evaluate the current behavior states and to detect an upcoming hazardous situation with a high probability. This knowledge has now to be used to initiate counter actions to avoid an upcoming drop-off or at least a highly critical situation. Because of the characteristics of that triggering, the robot is only able to react on these events if they are very close and if the robot has nearly reached the undesired state. In fact, the first chambers in driving direction have to enter the dangerous surface patch to analyze it. This will change if there exists a method to analyze the upcoming terrain in front of the robot via environmental sensors and to perform methods like near-to-far learning for classification. Due to limited payload and mounting space, suitable sensor systems do not exist so far which have the required resolution, accuracy and measuring area (compare section 2.1.5).

In mobile robotics a couple of general courses of action [Armbrust2007] exist which can be considered. Some of them can be used also in the range of climbing robots whereas others have to be adapted to this special application or cannot be applied. Section 9.2.1 for instance will introduce an emergency stop which is common for ground based vehicles but cannot be transfered directly to climbing machines. As in the case of the adhesion (section 7.5) or motion controllers (section 2.2.2), also these elements have been implemented as iB2C behaviors. The idea is to use the adhesion evaluation value $E$ as virtual sensor input to trigger the needed safety behavior.

9.2.1 Emergency Stop

The most common safety element in robotics is a full stop of the system in case of an emergency. The reactions range from an active braking to a power shutdown, if e.g. the control computer is no longer available (which can be detected via a watchdog unit on an embedded controller board) or a tactile sensor detects an impact. Robot manipulators in industrial applications are often positioned inside of a safety cage or a virtual cage which is defined by approximation sensors which detect if a human enters the robot’s surroundings. In this case the robot is stopped at the current position or its motion is slowed down. In general, an emergency stop can be implemented e.g. via a safety chain like in the indoor platform MARVIN which shuts down the motor power of the drive system if the chain is opened by a detected malfunction of one of the attached components [Hillenbrand2007a].

In the case of a climbing robot like CROMSCI a power shutdown via this kind of electronic circuit – also only of the drive system – would be disastrous since the suction engines as well as the drive units must stay powered to keep the robot adhered to the wall and to avoid an uncontrolled slipping down. Of course, also the valves, sensors, microcontrollers, and all other components for adhesion have to remain active. In this case other safety
measures have to be used. In fact, the emergency stop (ES) behavior has to be designed in a way that all motions of the drive system and of the manipulator arm have to be cancelled: Speed controllers (drive) have to reach a velocity value of zero, whereas position controllers (steering, manipulator) have to keep the current position. This safety measure is only triggered if the adhesion of the robot is endangered (depending on the adhesion evaluation) and not for collision avoidance. Therefore, this emergency stop can be supported additionally by a raising of the sealing pressure. In this case the sealing rubber inflates and is pressed strongly to the wall leading to a more leak-tight sealing. This allows the generation of a higher negative pressure and a recover of chamber and reservoir pressures to ensure robot adhesion. But on the other hand, a higher portion of downforce is absorbed by the strong inflated sealings which leads to a decreased downforce at the wheels. Therefore, robot navigation is not possible in this situation and the sealing pressure has to be reduced again to allow robot motion. Nevertheless, this measure gives the adhesion system a chance to recover the negative pressure of the leak-tight working chambers and to rebuild the reservoir pressure.

The emergency stop behavior has been implemented in a way that it allows a full stop as well as a controlled withdrawal of the triggered safety measures. Figure 9.9 shows the internal states $s_{ES}$ of the behavior:

**Inactive** If the behavior is in this state ($s_{ES} = -1$) it is either inhibited fully or not stimulated. This can be the case if it should not be used, if other more important measures are active, or if the user manually deactivates it to cancel the emergency stop.

**Ready** In case of an active behavior (activation $\iota_{ES} > 0$) the emergency stop starts operation ($s_{ES} = 0$), but it does not interfere with the other control elements yet. It surveys the trigger inputs $S_A$ (adhesion score) and $E$ (risk evaluation) for conspicuous values which make counteractive measures necessary. For this purpose, threshold values $\hat{E}^{tr}_{ES}$ and $\hat{S}^{tr}_{ES}$ have to be used which define the limits for triggering.

![Figure 9.9: Internal states of the emergency stop behavior with normal transitions (solid lines) and additional transitions in case of a deactivation of the behavior (dashed lines).](image-url)
9.2. Counteractive Measures

**Stop** This is the first phase of the emergency stop ($s_{ES} = 1$) if the trigger inputs are above certain thresholds. Now the robot is stopped at the current position and the sealing pressure is increased for leak tightness. In fact, it inhibits the behaviors which are responsible for locomotion via its general control output (see figure 9.11). Its own activity value is not suitable for inhibition since the motion control behaviors have to be inhibited selectively in the rescue phase.

**Rescue** Since a rope down of the robot should be the last option, the behavior switches to the second phase of emergency stop ($s_{ES} = 2$) if the adhesion system was able to recover and the adhesion score $S_A$ is below a threshold $S_{ES}^{min}$. Now the sealing pressure is reduced and the inhibitions of those motion behaviors are taken back which are in opposite direction to the last driving command. By this means, the operator can drive the robot manually away from the hazard. Afterwards – if $S_A$ is below a threshold $S_{ES}^{inactive}$ – also this safety measure is taken back and the behavior can switch to the ready state (if it did not become deactivated).

The meta data target rating $r_{ES}$ and activity $a_{ES}$ of the emergency stop behavior are calculated according to equations (9.7) and (9.8). The target rating is determined via the trigger inputs and thresholds as shown before whereas the activity depends on the internal state $s_{ES}$ of the behavior as depicted in figure 9.9.

\[ r_{ES} = \left\langle \max \left( \frac{E}{E_{ES}}, \frac{S_A}{S_{ES}} \right) \right\rangle^1 \quad (9.7) \]
\[ a_{ES} = \iota_{ES} \cdot \left( s_{ES} \right)^1 \quad (9.8) \]

Nevertheless, this emergency stop works only if the current position is more or less safe. If the robot is located on a large gap and several chambers have been deactivated, as described in section 7.3.4, the abidance at this position might not be the optimal solution. In fact, this safety measure can be applied if other measures fail and the robot is nearly falling off. In this case it is helpful if the robot is able to keep adhered and can be roped down manually and controlled. This rescue-by-hand also becomes necessary if the withdrawal of this measure fails and the robot is not longer able to remain adhered to the wall while the sealing pressure is lowered back to the normal level.

**9.2.2 Draw Back**

It should be obvious that the emergency stop behavior is not the tool of choice with respect to the operability of the robot system: First of all, it is not guaranteed that the stopping position is safe which is important for the functionality of this measure. Furthermore, it is also not very convenient if the robot has to be driven to a safe position by using joystick control or rescued manually via the security rope. The emergency stop remains to be the last fallback system but needs to be supported by a more practical safety measure.

In the present case a draw back (DB) behavior has been developed which replays the last motion commands in a reversed order. The idea is to return the same already negotiated path which was safe up to this point. This measure can be realized by storing the latest
Figure 9.10: Internal states of the draw back behavior with normal transitions (solid lines) and additional transitions in case of a deactivation of the behavior (dashed lines).

Driving commands in a history $H_{DB}$. Since it is important to leave the hazardous situation continuously without stopping, only the motion commands are recorded whereas periods without motion are neglected. Of course, it is sufficient to store only the commands of the last ten or twenty seconds so the history has been implemented as a kind of ring buffer. If the draw back behavior is triggered, the stored commands are played back with inverse motion direction until the system is safe again. This behavior is essential for robot safety to escape from critical situations and to avoid an upcoming drop-off of the robot. Figure 9.10 shows the four internal states which are similar to those of the emergency stop behavior:

**Inactive** As in the previous case of the emergency stop behavior, this behavior is inactive ($s_{DB} = -1$) if it is either inhibited from outside or not stimulated. This state is always entered if the internal activation $\iota_{DB}$ is zero. Nevertheless, the behavior collects motion data for its history $H_{DB}$.

**Ready** If the behavior becomes activated (activation $\iota_{DB} > 0$) it enters the ready state ($s_{DB} = 0$). Now it is armed and waits for the trigger event if the risk prediction value $E$ exceeds the threshold $\hat{E}_{tr_{DB}}$. During this time it still records motion commands.

**Running** In the running state ($s_{DB} = 1$) the behavior takes over the motion control of the robot by stimulating those motion behaviors which are responsible for drawing back (based on the recorded motion commands). Additionally, it inhibits the other behaviors to avoid that other behaviors may influence this escape trajectory. The behavior remains inside of this state until either the history has been processed completely so that no commands are left ($|H_{DB}| = 0$), or if both risk prediction value $E$ and adhesion score $S_A$ are below thresholds $\hat{E}_{DB}^{\max}$ or $\hat{S}_{DB}^{\max}$, respectively. Additionally, a minimum time $t_{DB}^{\min}$ of draw back motion can be set to ensure that the robot leaves the surface patch which caused this incident.

**Waiting** Afterwards ($s_{DB} = 2$), the robot stops at the current position and waits for a certain period of time $t_{DB}^{\min}$ or until the adhesion score is below a second threshold $\hat{S}_{DB}^{\infty}$ to ensure that the adhesion system was able to recover. During this time
all motion control behaviors are inhibited similar to the emergency stop behavior. Finally, the behavior switches to ready via the inactive state if the behavior still is activated.

The different meta data of the draw back behavior are calculated according equations (9.9) and (9.10). The target rating \( r_{DB} \) depends on the risk prediction value \( E \) and the number of motion commands \( |\mathcal{H}_{DB}| \) inside of the history compared to the maximal amount \( |\mathcal{H}_{DB}|_{\text{max}} \). As a result, the behavior is unsatisfied if the robot is endangered (high value of \( E \)) but the history is (nearly) empty so the system has no options to react. The activity \( a_{DB} \) only depends on the internal behavior state \( s_{DB} \) so it is active while it controls robot motion (running and waiting states).

\[
\begin{align*}
  r_{DB} &= \langle E - \frac{|\mathcal{H}_{DB}|}{|\mathcal{H}_{DB}|_{\text{max}}} \rangle_0 \quad (9.9) \\
  a_{DB} &= \iota_{DB} \cdot \langle s_{DB} \rangle^0 \quad (9.10)
\end{align*}
\]

This measure works well if it is activated soon enough since it strongly depends on a correct and early prediction of an upcoming hazard. Only the execution accuracy of the recorded motions may be problematic, e.g. if the robot has to drive the same way up the wall it just went down. Because of general wheel slip, the robot will drive a shorter distance upwards in the same period of time. But, this limitation is a general problem of locomotion and localization which has to be solved in the future. Assuming a working localization method, the draw back behavior can be improved in a way that the robot drives to the closest known safe position to avoid a drop-of and to recover the adhesion system. This should be the closest point to the current position which lies behind the robot. It can be determined by the robot itself by observing its internal state and risk estimation value and stored into a local map of the environment. Again, a correct robot localization is an indispensable requirement for this enhancement.

### 9.2.3 Integration into the Motion Control Network

Both safety behaviors – emergency stop and draw back – have been integrated into the motion control network which has been introduced in section 2.2.2. Figure 9.11 shows the basic behaviors (gray modules in the middle) for robot motion and corresponding fusion behaviors (blue modules). In the bottom left corner the emergency stop behavior is located which receives the trigger values \( E \) and \( S_A \) from the lower control elements inside of the robot control software. Additional inputs are the control values of the lowest motion fusion behaviors since the behavior needs to know which directions of motion must be inhibited to avoid further adhesion losses. As described in section 9.2.1, it calculates individual inhibition values for the motion behaviors to allow a selective blocking of special motions. In the same manner also the draw back behavior on top receives trigger values and the motion values as inputs. But here, the motion values are needed to create the history of robot motion as introduced earlier. If the behavior becomes active and starts the draw back, it generates individual motion commands based on the recorded history for the motion control behaviors which are given to the fusion elements below. Additionally,
it inhibits the *emergency stop* since it should be the only active safety measure at that time to avoid contradictory measures. Otherwise it would be possible that the robot faced a risky patch of surface so the *draw back* behavior becomes active and drives the robot backward. During that motion it might be possible that also the *emergency stop* becomes active which first lets the robot stop but later allows only a motion back to the hazard during the *rescue* phase (compare figure 9.9).

### 9.2.4 An Obsolete Driving Strategy

Nevertheless, also other methods to reduce the risk of a drop-off have been considered. Jens Wettach introduced another kind of driving strategy to overcome a deep straight horizontal crack without losing more than four chambers at once [Wettach2004]. This means, that at least three chambers can be evacuated and adhere the robot to the wall. Figure 9.12 illustrates the process of handling this crack. The idea is to rotate the robot if the center chamber and the two neighbors are located on the crack. Without this motion it would come to a loss of five chambers which is not sufficient for adhesion force. In the scope of this thesis this strategy has also been evaluated. Unfortunately, it cannot be applied in the present case because of tilt effects which have not been considered in the past. The problem is based on the distribution of downforces: Due to the fact, that the climbing robot has to balance out the downforces of all three wheels, it has to create a higher negative pressure at the chambers on top (chambers $C_1$, $C_2$ and $C_6$ in figure 9.12a).
and a pressure close to ambient air pressure at the bottom chambers \(C_3, C_4\) and \(C_5\). If the robot moves its frontal chambers on the crack, they lose their adhesion force and the robot is adhered only by the bottom chambers which have to create more negative pressure than before. But even if the downforce value itself would be high enough, the situation in figure 9.12c leads to a tilting of the robot since there exists no chamber on top which could be used to counteract the general torque of the robot (compare section 3.4.7 and the stability triangle in figure 3.14). Although this strategy cannot be applied here, it delivers important information on advantages and disadvantages of the mechanical setup of CROMSCI. One of these findings is that single adhesion chambers are more effective for handling cracks and that this strategy would work with another structure of the adhesion chambers or if the robot works overhead at a ceiling so that no tilting torque exists.

9.3 Experimental Results of Risk Prediction

Beside the experiments in the simulation environment to validate the general functionality of the presented approach of risk prediction some test with the real prototype CROMSCI have been executed. Due to the fact that this robot is in an early prototypic state, the number of test runs is very limited. Common hardware problems are leaky sealings, blocked wheel domes and wear of wheels. Within the range of a cooperation project these issues will be handled by several industrial partners, who are specialized on rubber technology, motors and controllers, or chassis construction.
9.3.1 Experimental Setup and Preparation

To prove the functionality of the approach under real conditions at first some training data is needed. As in the case of the simulation, these data sets have to be used to optimize the evaluation function by finding suitable weights, as introduced in section 8.1.2. For data collection and subsequent experiments a wooden test wall has been used. The advantages of this setup are the easy adaption of the wall (holes and grooves can be easily created) and its indoor location allowing it to test independently from weather. To test the system against defects several holes and one kerf have been drilled and milled. In a first step the robot has been driven sidewards several times until it reached the end of the wall (figure 9.14 illustrates this situation). At a particular point the robot is not able to stay adhered and it drops off. In total four training sets have been created: Three with a drop-off of the robot (secured by a rope) and one without, in which the end of the wall was not reached. The last one is important to avoid false alarms whereas the others are needed to get to know critical situations. Figure 9.13 depicts the developing of the adhesion score values $S_A$ of all training sets as gray graphs. In red the optimized evaluation values $E$ for risk prediction are shown. These evaluation values are based on the involved behavior values and the weights given in A.8.2, which have been found during the optimization process. Again, the goal is that the evaluation $E$ reaches a value above 1 one to two seconds before the adhesion score $S_A$ is at 1.

![Figure 9.13: Resulting evaluation values $E$ (red) based on the learned weights.](image_url)

Resulting risk estimation $E$ based on a training set without a robot drop off.

Results of three training sets with a robot drop off indicated by the adhesion score $S_A$. The evaluation values $E$ have been limited to -0.25 and 1.25 due to visualization properties.
9.3.2 Risk Prediction and Avoidance

The acquired evaluation function with the set of weights is now used to predict an upcoming drop-off if the robot is faced with similar situations during operation. Figure 9.14 shows some video stills of such an experiment. In this test run, the robot operator again tries to move the system over the edge of the wooden wall as it has been done before in the training phase. Due to the high dynamic of the system it can be stated that the robot is in a similar but not the same situation as before – so the evaluation is robust against some dissimilarities. Nevertheless, it was possible to record only a small set of training data and to perform a small number of validation runs\(^3\) due to hardware limitations caused by the large stress on the prototypic system.

\(^{3}\)Since only a handful of test runs could be executed a statistical examination has been omitted.

Figure 9.14: Video stills of experiments on wooden test wall (frontal and side view): The climbing robot CROMSCI detects an upcoming drop-off and avoids it by drawing back.
Figure 9.15 depicts the point of downforce $\vec{F}_{D}^{act}$ (top), the corresponding downforce value $F_{D|z}^{act}$ (middle), as well as the adhesion score $S_A$ and the risk estimation value $E$ (bottom). It can be seen that the downforce decreases at time step (A) ($t = 6.5$ s). This incident is related to the first chamber, which has been moved beyond the edge of the wall. The side view in figure 9.14 shows the motion and overlapping of the robot better than the frontal view since the chamber’s sealing lies some centimeters inside the robot chassis. At (B) the evaluation $E$ reached a value of 1, which activates the draw back behavior. 

![Figure 9.15: Experimental result on wooden test wall: Position and value of point of downforce (top, middle), adhesion score $S_A$, and estimated risk $E$ (bottom).](image)

This activity can be seen in figure 9.16. At the beginning the control via the graphical user interface (GUI) is fully active indicated by the green bar on top ($a_{\text{GUI}} = 1$) whereas the draw back (DB) behavior is activated (red bar) but not active ($\nu_{DB} = 1$, $a_{DB} = 0$). Both behaviors are able to stimulate the robot’s basic motions drive forward (F), drive backward (B), drive left (L), drive right (R), turn left, and turn right $^4$ (compare figure 9.11). The desired stimulation values are depicted in figure 9.17. It can be seen that the GUI stimulates forward and right driving nearly until the end. In contrast to that, the draw back behavior starts stimulation of oppositional motions backward and left at (B). In the behavior plots (figure 9.16) of the basic motions a switch of the motion direction can be observed. This is possible due to the fusion behaviors on top of the basic motions, which process the inputs from both control behaviors with different priorities. Therefore, the inputs from the GUI are overwritten by the ones from DB.

$^4$The turning behaviors are not shown here since the robot executed only straight motions during this experiment.
9.3. Experimental Results of Risk Prediction

At time step (C) all chamber areas returned to the wall and the adhesion force could be restored. Nevertheless, the draw back behavior continues its control (in this case: driving backwards and left) until adhesion score $S_A$ and risk estimation $E$ reached a secure level as shown in figure 9.15 at timestep (D). Now the DB behavior stops the robot at the current position and waits some seconds to allow the complete adhesion system to recover. Since

Figure 9.16: Experimental result on wooden test wall: Meta values of controlling behaviors GUI (user input) and draw back DB and of the four motion behaviors drive forward (f), drive backward (b), drive left (l) and drive right (r).

Figure 9.17: Experimental result on wooden test wall: Robot position (right) and stimulation values of the four basic drive motions (forward (f), backward (b), left (l), right (r)) via the controlling behaviors GUI and draw back DB (left).
the DB stays active (compare value of $a_{DB}$ at figure 9.16), the desired user inputs from the GUI are still overwritten by a full stop ($a_F = a_B = a_L = a_R = 0$). The position changes of the robot can be seen in figure 9.17 on the right side. Afterwards, the user commands are set to zero at (E), which deactivates the GUI behavior.

For completeness, also the chamber pressure values and valve positions of this test run are depicted in figure 9.18. Here it can be seen that the adhesion system tries to balance the robot’s downforce by generating a higher adhesion in chamber $C_1$ (dark red) lying on top than in the bottom chamber $C_4$ (dark blue). An interesting aspect is also the closing of valve $V_3$ (yellow) at (A) which reduces the total adhesion force by raising the chamber’s pressure. But this mechanism lowers the (horizontal) tilt in y-direction since chamber $C_3$ is the counterpart of chambers $C_5$ and $C_6$, which are located beyond the wall and lose negative pressure.

![Figure 9.18: Experimental result on wooden test wall: Individual chamber pressures (top) and valve areas (bottom).](image)

### 9.3.3 Safety Measures on a Concrete Wall

To test the prediction of risks and the corresponding safety measures under real conditions, the same experiment has also been executed on a concrete wall outside of a building. Figure 9.19 shows the videos stills of one test run. It can be seen that the system behaves in a similar way as in the previous example. For evaluation, the same set of weights has
been used, which has been trained on the wooden wall (see appendix A.8.2). This was possible because of the same situation, in which the robot is driven beyond the edge of the wall. The whole setup – driving beyond the wall’s edge – has been chosen again because it leads to a repeatable robot behavior (drop-off). Further tests are not possible due to limitations of the prototypic hardware.

\[ t = 0 \text{s} \]
\[ t = 2 \text{s} \]
\[ t = 4 \text{s} \]
\[ t = 6 \text{s} \]
\[ t = 8 \text{s} \]
\[ t = 10 \text{s} \]
\[ t = 12 \text{s} \]
\[ t = 14 \text{s} \]

**Figure 9.19:** Video stills of detection and evasion experiments on a concrete wall (frontal and side view).

Figure 9.20 depicts the development of the force values and the estimated risk. Although the estimated risk \( E \) is more irregular and with more peaks compared to the previous example (figure 9.15), it signals an upcoming drop-off sufficiently early to give the robot enough time for its counteractive measures. It can be expected, that the risk estimation is better if training examples of that particular situation are added to the optimization process to update the weights of the evaluation function.
Figure 9.20: Experimental result on a concrete wall: Position and value of point of downforce (top, middle), adhesion score $S_A$ and estimated risk $E$ (bottom).

All in all, the experimental results are very satisfying, although further tests are needed to validate the functionality of this approach in the field. In fact, more situations and different surface structures have to be tested to get to know the limitations of the system and whether it is possible to find weights, which are suitable to a broad kind of surfaces, or not.

9.4 Discussion of Risk Handling Measures

This chapter presented different approaches to handle risks for a climbing robot. Since the previous chapter only deals with requirement 15, which includes the detection of hazardous situations, this chapter focussed on the proper reaction to current hazards as postulated in requirement 16 in section 5.2.3.

At first, some techniques of obstacle detection and avoidance were introduced enabling the robot to meet two main safety requirements: The avoidance of obstacles (requirement 10) and a deceleration if objects are ahead (requirement 11). Through these two measures the climbing robot can avoid critical situations in terms of collisions and a driving into holes (section 3.4) which might cause severe damages at the robot structure, inspection device, or wheels and sealings. These situations can be avoided with a high probability if the robot localization has a certain degree of accuracy. Unfortunately, real-world experiments of these safety measures on a concrete wall could not be performed to a sufficiently large extent due to hardware limitations of the robot prototype.
The second part introduced counteractive measures related to the detection of hardly detectable surface features, which are suitable to escape from critical situations (draw back behavior) or to keep the robot adhered at the current position via an emergency stop behavior. These measures are of great importance to ensure robot safety since they avoid a drop-off of the system. In fact, they enhance robot adhesion and reduce the chance of robot tilt and slip, which are the main causes for a drop-off. In real-life experiments on a wooden test wall and on a concrete surface the functionalities of training data collection, optimization process, risk prediction, and risk avoidance via a corresponding safety measure could be shown. Of course, these experimental results using the real prototype CROMSCI are not comprehensive, but it could be shown that the selected approaches are viable under real conditions.
10. Conclusion

“Everything, that can work, will work.”

Yhprum’s Law [1]

Climbing on concrete structures is still a great challenge for mobile robots. Especially wheel driven systems using negative pressure adhesion are best suitable for inspection tasks on large concrete buildings due to a fast and continuous movement, but are always in danger of a drop-off. This thesis presented approaches to enhance navigation safety of such climbing robots based on a detailed analysis of hazards and possible error sources. It has been shown that low-level control elements are important to enhance safety although they are not sufficient to avoid a drop-off. In terms of hard predictable situations, higher online analysis methods and appropriate measures are mandatory for navigation safety – and to make the system work.

10.1 Summary

This thesis addresses the problem of safe navigation of wall-climbing robots by using analysis techniques and special measures. At the beginning, a detailed description of different hazards, which affect navigation safety of the robot, has been given. The main focus lies here on the prevention of the most dangerous incident: A drop-off of the robot. Based on a fault tree analysis several points of action were identified, which could be enhanced to increase the system’s safety. Main hazards for climbing robots like CROMSCI are robot slip and tilt, which lead to a drop-off, but also abrasion of wheel rubber and sealing coating, wheel slip, or collisions are serious incidents. These analytic results lead to several requirements being important for the safety of the system. Requirements 1 to 5 address the robot hardware and basic control components needed for operability. They are most important during development phase since changes in hardware as well as basic design decisions in software cannot be changed easily later on. The first three of

these requirements – fitting hardware components, sufficient reserves of supply elements, and robust controllers – have been fulfilled during the design and construction phase of the robot CROMSCI by using light-weighted materials, optimized and well-dimensioned components and carefully designed controllers. Requirement 4, which deals with the online identification of problems, has been handled via the special behavior-based control network used for closed-loop control and higher deliberative elements. This structure allows the application of visualization and debugging tools like MCA browser. The last of hardware-related requirement demands periodical check ups of high mortality parts. Since the prototype is still in development and not in final application, the operation time does not exceed the lifetime of critical elements like sealing or wheels.

![Some close-ups of CROMSCI’s hardware.](image)

**Figure 10.1:** Some close-ups of CROMSCI’s hardware.

Related to software elements used for safety enhancement, it can be distinguished between motion control components adjusting the locomotive capabilities and those, which are associated to the adhesion system. In the scope of this thesis a couple of techniques has been developed, which affect robot safety during locomotion. The kinematic calculations have been adapted to minimize wheel steering and to disallow steering on the spot, as demanded in requirements 6 and 9. Furthermore, a special traction control system has been developed on the DSPs to reduce wheel slip (requirement 7) based on transferable forces in wheel’s rolling direction. To reduce shear forces between the individual drive units (requirement 8) a unique shear force controller has been created. Both controllers affect the rotational speed and steering angle of the drive units in a way that the transferable force in rolling direction of each wheel is optimized. This significantly improves robot safety by reduced wheel slip and a reduced chance of robot slip, as shown in several real experiments. Additionally, the wear of the wheels and the load of the drive motors is decreased.

In the range of adhesion safety also several adaptive and controlling components have been developed which enhance the negative pressure system. At first, the complete closed-loop control system has been realized as a behavior-based network, which allows an easier
creation and adaption, but also a better debugging and tooling compared to classic controllers. This helps to analyze the current system state (requirement 4) and to identify problems or erroneous components. Beside the basic closed-loop control behaviors several further behaviors have been developed, which e.g. calculate the desired chamber pressures as demanded in requirement 12. In contrast to existing calculations using iterative methods, a new direct calculation has been presented, which considers chamber limits and works more accurate and faster. Furthermore, leak chambers are handled more efficiently to reduce the influence of large leakages on the overall adhesion system. In cases of large leakages or an upcoming collapse of the negative pressure, the corresponding chambers are cut off from the vacuum system immediately (requirement 13). To be conform to the structure of the behavior-based control network this is done by inhibiting the corresponding chamber behavior. For adhesion safety, it is important that these chambers are reintegrated into the adhesion system as soon as possible and without disturbing the remaining chambers (requirement 14). Therefore, several mechanisms have been developed, which test deactivated chambers periodically, cancel the testing immediately if a reactivation is not possible so far, or reintegrate chambers without testing if they are for sure leak-tight again. Experimental results show the advances of these measures resulting in shorter cut-off time, and better and more balanced adhesion control which is important for robot safety.

Beside the presented basic safety measures in terms of closed-loop control or parameter adjusting also the detection of actual and upcoming hazards has been handled. This is mandatory for navigation safety since there always exist situations in which the closed-loop controllers reach their limits and the system fails. These cases have to be detected and avoided to ensure the adhesion of the climbing robot, as demanded in requirements 15 and 16. In the scope of this thesis the developed behavior-based adhesion network is used to predict an upcoming drop-off by watching its behavioral meta data. Thus, an evaluation function has been created performing the online safety analysis. This function is learned based on training examples using a genetic algorithm, since the impact of the external influences on the system are not known and the behavior of the adhesion system can not be modelled sufficiently. Especially small surface irregularities and rough patches have an influence on the robot’s adhesion which can neither be described nor detected beforehand. It has been shown that it is possible to find evaluation functions for different situations, which allow a hazard prediction with high accuracy depending on the current direction of motion. Due to the fact that a foresighted detection via environmental sensors is not possible the only possible reaction on a detected hazard is drawing back and return to a known safe position. This safety measure limits the operability of the system but ensures its adhesion, which is most important. Its functionality and relevance for navigation safety has been demonstrated in several experiments. This includes several tests in the presented simulation environment as well as experiments with the real robotic prototype cromsci. Although only a few test runs could be executed due to the high stress on the robot’s hardware the results are very promising since the robot was able to detect a critical situation and evade it.

In cases of a foresighted detection other ways of risk handling are possible. This counts for obvious risks like large holes or protrude obstacles which can be detected e.g. via a laser range sensor. In the scope of this thesis also this kind of hazard has been considered, which includes the detection of large obstacles and corresponding counteractive measures.
A detection component has been developed using a light-weighted Hokuyo laser ranger, which delivers information about the area in front of the robot. Based on extracted points of interest, a local obstacle map is created which is used as virtual sensor for reactive components realized as motion behaviors. Depending on obstacle positions and motion direction, different behaviors become active, which slow down the robot or the manipulator arm – as demanded in requirement 11 – or turn the robot away from the hazard (requirement 10). The usage of a behavior-based motion control network allows further improvements of the robot’s reaction if e.g. more sensors are embedded into the robot platform. Experiments with the real prototype CROMSCI have shown that the system is able to detect and avoid risky obstacles to keep adhered to the surface and to prevent damages of the robot.

10.2 Discussion of Achievements

The shown combination of behavior-based closed-loop control, behavior survey, risk estimation and safety measures is completely novel in the context of climbing robots. As a result, the robot is able to detect critical situations leading to a drop-off and react on them. This aspect increases the overall safety of the robot tremendously.

Regarding the first and most important thesis demanding methods of risk estimation to detect dangerous situations and appropriate safety measures, several achievements could be accomplished. Again, simple closed-loop controllers are not sufficient for navigation safety since they reach their limits in certain situations causing a robot drop-off. The presented online safety analysis in terms of a risk prediction function is able to detect trained risks with an accuracy rate of 0.961 (96.8% true positives, 5.9% false positives, compare section 8.4.3). This is done by a survey of the internal states and meta data of adhesion control behaviors interacting in a complex network. The analysis proposed in this work reveals new safety information which could not be retrieved so far and are important to initiate counteractive measures. As stated in thesis 2, the control structure has to support the analysis of the system’s state by providing information about the internal procedures. This aspect is handled by the behavior-based network for adhesion control with individual components sharing meta data and further values. Due to the fact that the network elements have a common interface, it is possible to append new behaviors to the network. In the same way the presented approach can be used to analyze a completely different network – based on an individual evaluation function. The gained information about the system’s state is then used to detect hazardous situations, which would lead to a drop-off and to initiate e.g. a draw back of the robot to a safe position. This mechanism prevents it from falling down since the closed-loop controllers are not able to handle this situation. The utilizability of the approach has been shown in the simulation environment as well as on the real prototype CROMSCI.

Further methods have been developed reducing risks that the robot has to handle constantly during normal operation. This addresses locomotion components as well as adhesion elements and supports thesis 3 demanding safety measures on all stages of control, which includes high-level control strategies in the same way as basic closed-loop controllers. The advantages of the presented advanced motion control (AMC) for robot safety can be summed up as follows: Increased lifetime of wheel rubber (less wheel slip), of wheel mechanics (lower sideward forces), and of drive motors (lower permanent current).
Especially the reduced wheel slip and better transfer of holding and motion forces causes a decreased chance of robot slip which is important for safety. Experiments have proven that shear forces and needed friction coefficient could be reduced by the SFC up to 59.6% and 21.0%, respectively. Also the TCS reduces the needed friction coefficient and allows a faster robot motion, which reduces the time in operation. The second type of control elements increase overall safety by enhancing robot’s adhesion via a more balanced adhesion and faster adaptions on the current situations. It could be shown that in critical situations the difference between current and desired downforce reached only 390 N compared to up to 1027 N, which is unacceptable.

All these measures contribute to navigation safety and are important for the overall safety of the robot. Nevertheless, they can only be set up and developed by a careful look at the system components and an offline analysis of the tasks, hazards, and the interaction of the system with the environment, as demanded in thesis 4. Since a chain is only as strong as its weakest link, an analysis is important to find the most important points of action. As described, it does not make sense to enhance components with a low failure rate if other malfunctions or problems occur much more often.

The contributions presented in this thesis can be summed up as follows:

- An in-depth analysis of hazards, related to climbing robots using negative pressure adhesion in combination with a drive system, and their causes,
- a novel traction control system to reduce wheel slip especially in vertical direction,
- an unique controller to reduce shear forces, which work contrary to the motion direction of the robot,
- an algorithm to calculate individual adhesion chamber forces based on an overall downforce, which takes individual force limits into account,
- an optimized strategy of chamber deactivation and reintegration in cases of high leakages,
- a novel combination of hierarchical closed-loop control elements and a behavior-based network,
- a risk prediction method based on the internal robot state by using evolutionary computation to optimize an evaluation function,
- and detection components to identify obvious risks for such climbing robots and reactive measures and strategies to avoid detected hazards.

After all, it still can be stated that – according to Murphy’s Law – “anything that can go wrong, will go wrong”\(^2\). But, in fact the probability of occurrence of these undesired events, reaching from wheel wear to robot tilt, is reduced to a minimum by the presented approaches.

\(^2\)http://en.wikipedia.org/wiki/Murphy%27s_law
10.3 Future Perspectives

So far, the findings of this thesis allow it to navigate a climbing robot on the wall semi-autonomously. But, to bring the system into application, it is not sufficient to react on actual events if the robot has already reached a critical situation. The system has to be able to predict risks of the upcoming terrain or of its next actions. Therefore, an online prediction system is needed to link existing knowledge of the environment and its risks with unexplored terrain. As pointed out before, this problem is not easy to solve since there neither exist suitable environmental sensors nor a model describing the influence of different surface features on the adhesion system. Whereas the second aspect might be handled via near-to-far learning methods, as e.g. applied by [Stavens2006], the sensors seem to be the bigger problem. Currently a diploma thesis is in progress which deals with these aspects and evaluates in the simulation environment what resolution of a 3D-sensor – here a simulated stereo-camera is used – is needed to detect necessary surface features. Furthermore, a set of surface and volume measures are evaluated whether they contain important information to predict e.g. the sealing’s leakage at these positions. So far, support vector machines (SVM) with a sufficient number of training sets seem to be a suitable method to find surface measures for the prediction of the behavior of the adhesion system. The final goal is to get a relationship between environmental features and risk prediction value. This combination allows it to remain adhered during navigation because of the prediction and reactive components, but also to collect and – later on – use data from external sensors to enable a foresighted navigation. Additional software measures are needed to react in a correct manner on the detected features lying ahead to ensure safety in the same way as the operability of the system.

Apart from the environmental sensing, the risk estimation procedure would also benefit from online training in a way that the robot might collect additional training data during normal operation. Of course it would not be possible to get examples for false alarms (false positives) and correct detections (true positives) since the robot system will react on the detected hazard and evade it – so the validation, whether the robot would have been fallen down or not, is missing. But, the optimization system gets important training sets related to failed detections (false negatives) and uncritical surfaces (true negatives). These training examples have to be added to the existing ones and used to update the current set of weights until all – old and new – training sets are covered by these weights. Nevertheless, the obtained research results generally would benefit from additional training sets collected by the real prototype and from long-term experiments. However, these tests impose high stress on the robot prototype, especially on its high mortality parts.

Regarding the experimental platform CROMSCI further modifications towards a system, which can be applied for the desired inspection and maintenance tasks, mainly address enhancements of the existing hardware and the integration of new components. In fact, the risk of a robot drop-off can be reduced by decreasing the robot’s weight. As an effect, less forces would be needed to keep the robot adhered to the wall what also decreases wheel slip (since the wheels have to transfer less forces) and increases the lifetime of the wheel’s rubber. Also the probability of robot tilt can be minimized by moving the wheels outside of the adhesion system so that all adhesion chambers lie within the wheel’s triangle. But, this change comes with the disadvantage that the robot would increase in its size what makes it less handy by the operator. At the present time, a new prototype is in progress with optimized characteristics concerning hardware structure, weight, adhesion system,
sealing material [Schmidt2012a] and further components. Since the software elements of CROMSCI have been developed in a modular way a transfer from the existing modules to a new robot control network is very easy. The same counts for the risk prediction method which is scalable since only new training data sets have to be acquired to find a suitable evaluation function. Only the adhesion control system has to be extended since the next prototype will use eleven negative pressure chambers. The design and development process of this robot strongly benefits from the presented analysis, safety measures, control strategies and risk prediction. Without these aspects the development would be much more cost-intensive and time consuming. On a long term, the results of this thesis will contribute to develop more efficient and safe wall-climbing robots, which will be able to fulfill different tasks at a large variety of buildings.
10. Conclusion
A. Appendix

A.1 Mathematical Notation

The notation of variables follows the style as given in equation (A.1)

\[ [c] X^Z_{Y|y} \]  \hspace{1cm} (A.1)

with

- \( X \): Identifier of the (physical) value, e.g. force, pressure or area
- \( Y \): Belonging of this value, e.g. negative pressure system or chamber
- \( y \): Precise description of the value, e.g. direction of the force
- \( Z \): Special values, e.g. minimum, maximum or current value
- \( c \): Coordinate frame, e.g. environment or robot

An example for this notation is \( F_{N|z}^{max} \) which denotes the maximum (\( max \)) force (\( F \)) in \( z \)-direction (\( z \)) of the negative pressure system (\( N \)). Table A.1 lists the used identifiers, their meaning and the used physical unit. A list of belongings can be found in table A.2 for general robot components.

The descriptions and specializations should be self-explaining since they just give more detailed informations e.g. about the direction of a force (\( x, y, z \)) or denote a minimum (\( min \)), maximum (\( max \)) or the current (\( act \)) value. The corresponding coordinate frame again is taken out of the pool of belongings as given in table A.2 so that e.g. \( [R] \vec{P}_H \) is the position (\( P \)) of the Hokuyo laser scanner (\( H \)) in robot coordinates (\( R \)) whereas \( [E] \vec{P}_H \) is the position related to the environment \( E \).
Table A.1: List of used identifiers

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Meaning</th>
<th>Unit</th>
<th>Identifier</th>
<th>Meaning</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Area</td>
<td>[m$^2$]</td>
<td>r</td>
<td>Radius</td>
<td>[m]</td>
</tr>
<tr>
<td>d</td>
<td>Distance / Dimension</td>
<td>[m]</td>
<td>$\mathcal{S}$</td>
<td>Set</td>
<td>-</td>
</tr>
<tr>
<td>F</td>
<td>Force</td>
<td>[N]</td>
<td>s</td>
<td>State</td>
<td>-</td>
</tr>
<tr>
<td>h</td>
<td>Height</td>
<td>[m]</td>
<td>t</td>
<td>Time</td>
<td>[s]</td>
</tr>
<tr>
<td>I</td>
<td>Motor current</td>
<td>[A]</td>
<td>$T$</td>
<td>Temperature</td>
<td>[K]</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>Amplification factor</td>
<td>-</td>
<td>u</td>
<td>Update value</td>
<td>-</td>
</tr>
<tr>
<td>l</td>
<td>Length</td>
<td>[m]</td>
<td>v</td>
<td>Velocity</td>
<td>[m/s]</td>
</tr>
<tr>
<td>m</td>
<td>Mass</td>
<td>[kg]</td>
<td>V</td>
<td>Volume</td>
<td>[m$^3$]</td>
</tr>
<tr>
<td>n</td>
<td>Number / Index</td>
<td>-</td>
<td>w</td>
<td>Weighting</td>
<td>-</td>
</tr>
<tr>
<td>p</td>
<td>Pressure</td>
<td>[Pa]</td>
<td>x</td>
<td>X Position</td>
<td>[m]</td>
</tr>
<tr>
<td>$\vec{P}$</td>
<td>Position</td>
<td>[m$^n$]</td>
<td>y</td>
<td>Y Position</td>
<td>[m]</td>
</tr>
<tr>
<td>pr</td>
<td>Priority value</td>
<td>-</td>
<td>z</td>
<td>Z Position</td>
<td>[m]</td>
</tr>
</tbody>
</table>

Table A.2: List of possible belongings

<table>
<thead>
<tr>
<th>Belonging</th>
<th>Meaning</th>
<th>Belonging</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Adhesion engine</td>
<td>$M_i$</td>
<td>Map sector $i$</td>
</tr>
<tr>
<td>C</td>
<td>Chamber(s) in general</td>
<td>$N$</td>
<td>Negative pressure system</td>
</tr>
<tr>
<td>$C_i$</td>
<td>Chamber $i$</td>
<td>$O$</td>
<td>Obstacle</td>
</tr>
<tr>
<td>D</td>
<td>Drive system</td>
<td>$P$</td>
<td>Point(s) from scanner</td>
</tr>
<tr>
<td>E</td>
<td>Environment</td>
<td>$P_i$</td>
<td>Point $i$</td>
</tr>
<tr>
<td>H</td>
<td>Hokuyo laser ranger</td>
<td>$R$</td>
<td>Robot</td>
</tr>
<tr>
<td>I</td>
<td>Inspection system</td>
<td>$S$</td>
<td>Sealing</td>
</tr>
<tr>
<td>L</td>
<td>Leakage(s) in general</td>
<td>$V$</td>
<td>Valve(s) in general</td>
</tr>
<tr>
<td>$L_i$</td>
<td>Leakage of chamber $i$</td>
<td>$V_i$</td>
<td>Valve of chamber $i$</td>
</tr>
<tr>
<td>$L_{ij}$</td>
<td>Leakage between $C_i$ and $C_j$</td>
<td>$W$</td>
<td>Wheel(s) in general</td>
</tr>
<tr>
<td>M</td>
<td>Map sector in general</td>
<td>$W_i$</td>
<td>Wheel $i$</td>
</tr>
</tbody>
</table>

A.1.1 Special Notations

Vector $\vec{X}$: Multi-dimensional vector of type $X$, which is used in most cases in combination with position descriptions.

Threshold $\hat{X}$: Threshold of type $X$, which can be specified by upper characters to distinguish e.g. minimum and maximum or upper and lower thresholds.

Difference $\Delta X$: A difference value of type $X$. This is no absolute value and can be used in conjunction with other difference values or as an offset.

Estimation $\hat{X}$: This is an estimation value of type $X$, which cannot be calculated or described exactly.

Derivations $\dot{X}$, $\ddot{X}$: Denotes the first and second derivation of a value of type $X$. 
A.1.2 General Mathematic Functions

Angular distance Equation (A.2) describes the absolute shortest angular distance of two angles $\alpha, \beta \in [-\pi, \pi)$:

$$|\alpha, \beta|^\circ = \begin{cases} 
\frac{|\alpha - \beta|}{2\pi} & \text{if } |\alpha - \beta| \leq \pi \\
2\pi - |\alpha - \beta| & \text{else }
\end{cases} \quad (A.2)$$

Domains The following notation of domains or codomains of functions are used:

- $[a, b]$ ranges from $a$ to $b$ including both values.
- $(a, b]$ ranges from $a$ to $b$ including only $b$.
- $(a, b)$ ranges from $a$ to $b$ excluding both values.

Limit The following notation (equation (A.3)) is a shortcut for the limit function using an upper $u$ and lower limit $l$ and the input value $a$:

$$\langle a \rangle_l^u = \begin{cases} 
l & \text{if } a < l \\
u & \text{if } a > u \\
a & \text{else}
\end{cases} \quad (A.3)$$

Modulo The modulo operation, which delivers the remainder of a discrete division:

$$a \mod b = a - \left\lfloor \frac{a}{b} \right\rfloor \cdot b \quad (A.4)$$

A.1.3 Statistics

For a validation of the experimental results (compare section 8.4.3), some statistical measures are used. In this context, some counted measurands are important:

- $\#tp$ is the number of true positives, which means that the test result was true and the candidate identified as positive.
- $\#fn$ is the number of false negatives. Here the test result was false and the candidate was not identified as positive.
- $\#fp$ is the number of false positives. The test identified the candidate as positive although it is negative.
- $\#tn$ is the number of true negatives. The candidate is negative and the test identified this in a correct way.
The following list gives an overview on the measures and on their definitions:

**Sensitivity**  
*Sensitivity* is also known as *true positive rate* or *hit rate* and describes the amount of positive classifications compared to the total number of elements which need to be classified positive:

\[
T_p = \frac{\#tp}{\#tp + \#fn} \quad (A.5)
\]

**Miss rate**  
This *false negative rate* is the counterpart of *sensitivity* and is the rate of wrong classifications of positive elements:

\[
F_n = \frac{\#fn}{\#tp + \#fn} = 1.0 - T_p \quad (A.6)
\]

**Specificity**  
*Specificity* – also known as *correct rejection rate* or *true negative rate* – describes the proportion of correct negative classifications compared to the total number of negative elements:

\[
T_n = \frac{\#tn}{\#tn + \#fp} \quad (A.7)
\]

**Fallout**  
*Fallout* or *false positive rate* is the proportion of false alarms:

\[
F_p = \frac{\#fp}{\#tn + \#fp} = 1.0 - T_n \quad (A.8)
\]

**Accuracy**  
The *accuracy* describes the total rate of correct classifications:

\[
T_c = \frac{\#tp + \#tn}{\#tp + \#tn + \#fp + \#fn} \quad (A.9)
\]

### A.1.4 Matrix Operations

**Rotation matrix in 2D**  
Rotation matrix which is used to rotate a point in 2D space around the given angle \( \alpha \).

\[
R(\alpha) = \begin{pmatrix}
\cos(\alpha) & -\sin(\alpha) \\
\sin(\alpha) & \cos(\alpha)
\end{pmatrix} \quad (A.10)
\]
Rotation matrices in 3D

Single rotation matrices in three-dimensional space around the different axes with angle $\alpha$.

\[ R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix} \]  
(A.11)

\[ R_y(\alpha) = \begin{pmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{pmatrix} \]  
(A.12)

\[ R_z(\alpha) = \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \]  
(A.13)

A.1.5 General Evaluation Functions

This section describes some general functions, which might be used for evaluation instead of the chosen one (compare section 8.1.1). While some of the proposed functions like the maximum or weighted mean are not applicable because of the limited results, others might be impossible to use because of the large degree of freedom. For generality, the presented evaluation methods use an universal input vector $\vec{x}$.

**Maximum function** This function $F_{E_1} : [0,1]^n \mapsto [0,1]$ (equation (A.14)) uses the chosen $n \in \mathbb{N}$ input values (e.g. activity and target rating values or variations of them). Unfortunately, there must exist a single identifier for the evaluation, and even if it exists, it might be unknown or does not reach a value of 1 at its maximum.

\[ F_{E_1}(\vec{x}) = \max_{i=1}^{n} (x_i) \]  
(A.14)

**Weighted maximum function** The weighted maximum function $F_{E_2}(\vec{x}) : [0,1]^n \mapsto \mathbb{R}$ with weights $w_i \in \mathbb{R}$ as given in equation (A.15) is similar to the maximum function, but allows the value to reach 1. Nevertheless, the problem of the missing single identifier is still existing so that also in most other applications these maximum functions do not seem to be sufficient. Additionally, suitable weights $w_i$ for the chosen set of input values are needed.

\[ F_{E_2}(\vec{x}) = \max_{i=1}^{n} (w_i \cdot x_i) \]  
(A.15)

**Weighted mean** This function $F_{E_3} : [0,1]^n \mapsto [0,1]$ (equation (A.16)) is very similar to the applied weighted sum (section 8.1.1). In this case the weighted sum is divided additionally by the sum of all weights. But, the weights are limited to positive numbers only ($w_j \in \mathbb{R}^+$) so that $w_j \geq 0.0 \ \forall \ j \in \{1, 2, ..., n\}$. In general, this evaluation function seems to be as powerful as the applied one, but it comes with a couple
of restrictions. First of all, each element can have only positive influence on the result since negative weights cannot be used. Furthermore, it is nearly impossible to reach an evaluation values of 1 since the weighted sum is divided by the sum of weights. Finally, this division also leads to multiple sets of weights which deliver the same results. Of course, this approach could be used in the present case, although it might not be the optimal function for evaluation.

\[ F_{E_3}(\vec{x}) = \frac{\sum_{i=1}^{n} (w_i \cdot x_i)}{\sum_{i=1}^{n} (w_i)} \]  

(A.16)

**Boolean expressions** The evaluation can also be limited to a logical value true or false via function \( F_{E_4}(\vec{x}) : [0,1]^n \mapsto \{0,1\} \) using boolean expressions to differ safety-relevant from irrelevant situations. Equation (A.17) shows the calculation of that value depending on the input vector \( \vec{x} \). This function applies \( m \in \mathbb{N} \) boolean expressions. Each of these expressions makes use of at least one value \( x_i \) from the input vector and a second value for comparison which can be either another input value or a real value with \( y_j \in \{x_0, x_1, ...\} \cup \mathbb{R} \). The operator \( \odot_j \in \{<, \leq, =, \geq, >, \neq\} \) can be chosen out of a general set of comparisons. Finally, all expressions are combined via logical operators \( \odot_j \) which also include concatenations of operators, brackets or similar elements to create any kind of boolean expression as given in extended Backus–Naur form (EBNF) in equation (A.17).

\[ F_{E_4}(\vec{x}) = \bigotimes_{j=1}^{m} (x_i \odot_j y_j) \]  

(A.17)

with \( \bigotimes_{j=1}^{m} z_j = z_1 \odot_2 z_2 \odot_3 z_3 \cdots \odot_m z_m \)

and \( \odot_j = [\land] \lor [\neg] \lor [\neg \land] \lor [\neg \land \lor] \lor [\neg \land \lor \land] \);

It is obvious, that this evaluation function is powerful and might be able to deliver the desired values. The main problem is of course the determination of the total number of needed elements \( m \), the logical operators \( \odot_j \) as well as the values \( y_j \) and operators \( \oplus_j \) for comparison. Due to the large amount of possibilities and the missing system knowledge they cannot be set by hand and need to be determined via an optimization or learning approach. Nevertheless, the output of this evaluation function is only true or false which seems to be insufficient to describe a safety state in general.

**Generic function** A very variable method for evaluation is a generic function \( F_{E_5}(\vec{x}) : [0,1]^n \mapsto \mathbb{R} \), which concatenates input values and weights with basic arithmetic operations. Therefore, equation (A.18) combines \( m \) values with \( y_j \in \{x_0, x_1, ...\} \cup \mathbb{R} \) via arithmetic operations \( \oplus_j \) which can be chosen out of a general set of operators including brackets as given in EBNF in equation (A.18). Of course, it is also possible to add mathematic functions like sinus or cosinus.
\[ \mathcal{F}_{E_\alpha}(\vec{x}) = \bigoplus_{j=1}^{m}(y_j) \]  
(A.18)

with \[ \bigoplus_{j=1}^{m} z_j = z_1 \oplus z_2 \oplus z_3 \cdots \oplus z_m \]
and \[ \oplus_j = \left[ \left( + \ | \ - \ | \ \cdot \ | \ / \right) \right] ; \]

This method is a more general approach than the weighted sum from equation (8.1), which can be created by this generic function, too. It is much more powerful since it allows a couple of more interaction between the input values like multiplication or division. Nevertheless, the amount of possibilities increases a lot since not only the set of used values \( y_j \) including the input data \( x_j \) and optional weights, offsets or multiplication factors have to be found. Also the total amount of values \( m \) and the arithmetic operations have to be selected. This increases the search space tremendously which would be hard to optimize within a sufficient short time.

**Case-dependent generic function** The most flexible function for evaluation would be a mixture of weights, mathematical operations, comparisons and logical operations. This case-dependent generic function \( \mathcal{F}_{E_\alpha} : [0, 1]^n \rightarrow \mathbb{R} \) would be most powerful and able to describe any desired function – if it can be described. Equation (A.19) is a combination of generic functions which are used or neglected depending on the result of the corresponding boolean expression with \( z_{kj} \in \{ x_0, x_1, \ldots \} \cup \mathbb{R} \).

\[
\mathcal{F}_{E_\alpha}(\vec{x}) = \bigoplus_{j=1}^{m} \left[ \bigcirc_{k=1}^{o_j} (x_i \oplus_{k_j} y_{kj}) \cdot \bigoplus_{k=1}^{p_j} (z_{kj}) \right] \]  
(A.19)

Although this function is most powerful, it should be obvious that the enormous amount of possibilities is nearly impossible to handle, even with highly sophisticated optimization algorithms.
A.2 Symbols

A summary of the symbols for hardware components, which are used in several figures inside of this thesis, is given in table A.3. Furthermore, this table shows common control symbols as they are applied in the field of control engineering.

**Table A.3:** List of hardware (left) and controller symbols (right).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Hardware component</th>
<th>Symbol</th>
<th>Control element</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Symbol" /></td>
<td>Air pressure sensor</td>
<td><img src="image2" alt="Symbol" /></td>
<td>Dirac impulse</td>
</tr>
<tr>
<td><img src="image3" alt="Symbol" /></td>
<td>Drive of a wheel unit</td>
<td><img src="image4" alt="Symbol" /></td>
<td>Integral component</td>
</tr>
<tr>
<td><img src="image5" alt="Symbol" /></td>
<td>General motor</td>
<td><img src="image6" alt="Symbol" /></td>
<td>Limit</td>
</tr>
<tr>
<td><img src="image7" alt="Symbol" /></td>
<td>Laser range sensor</td>
<td><img src="image8" alt="Symbol" /></td>
<td>Proportional component</td>
</tr>
<tr>
<td><img src="image9" alt="Symbol" /></td>
<td>Load cell</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image10" alt="Symbol" /></td>
<td>Manipulator arm</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image11" alt="Symbol" /></td>
<td>Motor encoder</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image12" alt="Symbol" /></td>
<td>Steer of a wheel unit</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image13" alt="Symbol" /></td>
<td>Valve of an adhesion chamber</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### A.3. Key Data

Table A.4: List of CRomSci’s key data, physical constants, and parameters of the advanced motion control elements.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{C_{1,6}}$</td>
<td>Area of outer chambers $C_1$ to $C_6$</td>
<td>0.057596 m$^2$</td>
</tr>
<tr>
<td>$A_{C_7}$</td>
<td>Area of center chamber $C_7$</td>
<td>0.061275 m$^2$</td>
</tr>
<tr>
<td>$A_T$</td>
<td>Total area of negative pressure system</td>
<td>0.40715 m$^2$</td>
</tr>
<tr>
<td>$A_V^{\text{max}}$</td>
<td>Maximum valve area</td>
<td>0.0003 m$^2$</td>
</tr>
<tr>
<td>$l_W$</td>
<td>Wheel distance from robot center</td>
<td>0.26 m</td>
</tr>
<tr>
<td>$r_W$</td>
<td>Wheel radius</td>
<td>0.053 m</td>
</tr>
<tr>
<td>$d_{W</td>
<td>y}$</td>
<td>Wheel width</td>
</tr>
<tr>
<td>$r_S$</td>
<td>Sealing radius</td>
<td>0.36 m</td>
</tr>
<tr>
<td>$h_S^{\text{max}}$</td>
<td>Sealing indentation (maximum)</td>
<td>$\approx$ 0.006 m</td>
</tr>
<tr>
<td>$h_R$</td>
<td>Height of robot chassis</td>
<td>0.4 m</td>
</tr>
<tr>
<td>$r_R$</td>
<td>Radius of robot chassis</td>
<td>0.4 m</td>
</tr>
<tr>
<td>$w_R$</td>
<td>Weight of robot</td>
<td>50 kg</td>
</tr>
<tr>
<td>$w_I$</td>
<td>Weight of inspection sensors</td>
<td>10 kg</td>
</tr>
<tr>
<td>$\rho_{\text{air}}$</td>
<td>Density of air</td>
<td>1.1883 kg/m$^3$</td>
</tr>
<tr>
<td>$\kappa_{\text{air}}$</td>
<td>Adiabatic exponent of air</td>
<td>1.402</td>
</tr>
<tr>
<td>$R$</td>
<td>Ideal gas constant</td>
<td>287.058 J/(kg K)</td>
</tr>
<tr>
<td>$T_{\text{air}}$</td>
<td>Temperature of air</td>
<td>293.15 °K (20 °C)</td>
</tr>
<tr>
<td>$I_p$</td>
<td>Update factor to raise PWM value</td>
<td>3.0</td>
</tr>
<tr>
<td>$I_m$</td>
<td>Update factor for limitation of PWM value</td>
<td>50.0</td>
</tr>
<tr>
<td>$\hat{I}_{\text{low}}$</td>
<td>Lower motor PWM threshold of drive motor</td>
<td>800.0</td>
</tr>
<tr>
<td>$\hat{I}_{\text{up}}$</td>
<td>Upper motor PWM threshold of drive motor</td>
<td>20000.0</td>
</tr>
<tr>
<td>$\hat{\mu}_{\text{stat}}$</td>
<td>Estimated friction value rubber-concrete$^a$</td>
<td>0.8</td>
</tr>
<tr>
<td>$K_{SFC</td>
<td>v,P}$</td>
<td>Amplification parameter of driving SFC</td>
</tr>
<tr>
<td>$K_{SFC</td>
<td>v,I}$</td>
<td>Integral parameter of driving SFC</td>
</tr>
<tr>
<td>$K_{SFC</td>
<td>\phi,P}$</td>
<td>Amplification parameter of steering SFC</td>
</tr>
<tr>
<td>$K_{SFC</td>
<td>\phi,I}$</td>
<td>Integral parameter of steering SFC</td>
</tr>
</tbody>
</table>

$^a$ The estimated friction value has been set according to literature and verified by experiments with the real robot. But, of course, it needs to be adapted depending on the individual combination of wheel and surface materials for optimal results.
A.4 Detailed Derivations

This section contains some detailed calculations and derivations, which would have gone beyond the scope of the main chapters. Cross references inside of the corresponding sections point on the following descriptions concerning kinematic constraints, roll and tilt, or the calculation of chamber forces.

A.4.1 Kinematic Constraints

Figure A.1 shows parameters, which describe the setup of a steerable standard wheel with distance $d$ from the robot center, angle $\alpha$, which describes its position relative to it and wheel steering angle $\beta$. The angle $\varphi$ corresponds to the wheel steering relative to the robot x-axis as it is used for kinematic calculations (compare section 2.1.2).

The application of rolling and sliding constraints for each wheel as presented by Siegwart et al. [Siegwart2004] in combination with the robot setup leads to the equation system (A.20). It has to be kept in mind that $\alpha_{Wi}$ defines the angle between the normal vector of the plane of wheel $i$ to the x-axis of the robot coordinate frame (compare figure A.1). These three angles are fixed depending on the robot setup and can be determined according to figure 2.4. The steering angle $\beta_{Wi}$ defines the angle between the straight line through kinematic center and wheel center point and the y-axis of the wheel coordinate frame. Those three angles $\beta_{Wi/2/3}$ are the control parameters to determine the direction of the wheel, which are set fix if the standard wheel is not steerable. Furthermore, $d_{Wi}$ is the distance between wheel $Wi$ and kinematic center, and $v_{Wi} = \omega_{Wi} \cdot r_{Wi}$ is the linear wheel velocity depending on rotational velocity $\omega_{Wi}$ and wheel radius $r_{Wi}$. The steering angles $\beta_{Wi}$ with $i \in \{1, 2, 3\}$ can be determined by using the sliding constraints whereas the rolling constraints are used for the calculation of the rotational velocities according to section 2.1.2.
A.4. Detailed Derivations

Without loss of generality it can be assumed that the frontal wheel whereas the rear part of the sealing is impressed. The trafficability of small obstacles and flat deepenings does not only depend on the ground clearance of a robot. These disturbances also cause the robot to roll and tilt which influences the sealing’s indentation (compare figure 3.9 in section 3.4.4). If the front wheel is positioned on a small step, the resulting robot tilt lets the frontal sealing expand whereas the rear part of the sealing is impressed.

Without loss of generality it can be assumed that the frontal wheel $W_1$ has no vertical offset and is located at $\vec{P}_{W_1} = (0.0, 0.0, 0.0)^{-1}$ whereas both rear wheels $W_2$ and $W_3$ have height values of $h_{O_2}$ and $h_{O_3}$ because of obstacles or holes $O_2$ and $O_3$. According to the drive setup as shown in section 2.1.1 with a distance $l_W$ of the wheels to the robot center robot, roll $\phi_R$ and tilt $\theta_R$ can be calculated. This can be done by equalizing the rotated $z$-positions of the rear wheels and the obstacle heights. As shown in equations (A.21) and (A.22), a rotation of the basic wheel positions $\vec{P}_{W_2}$ and $\vec{P}_{W_3}$ is done according to roll-pitch-yaw and set equal to the new coordinates with the given height values.

\[
\begin{align*}
\vec{P}_{W_2}^* & = \begin{bmatrix} x'_{W_2} \\ y'_{W_2} \\ h_{O_2} \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} R_z(\theta_R) \cdot R_y(\phi_R) \\ \cos \theta_R & \sin \theta_R \cdot \sin \phi_R & \sin \theta_R \cdot \cos \phi_R \\ 0 & -\cos \phi_R & -\sin \phi_R \\ -\sin \theta_R & \cos \theta_R \cdot \sin \phi_R & \cos \theta_R \cdot \cos \phi_R \end{bmatrix} \cdot \begin{bmatrix} x_{W_2} \\ y_{W_2} \\ h_{O_2} \end{bmatrix} = \begin{bmatrix} R_z(0) \cdot R_y(\phi_R) \\ 0 & -\cos \phi_R & -\sin \phi_R \\ -\sin \theta_R & \cos \theta_R \cdot \sin \phi_R & \cos \theta_R \cdot \cos \phi_R \end{bmatrix} \cdot \begin{bmatrix} x_{W_2} \\ y_{W_2} \\ h_{O_2} \end{bmatrix} = \begin{bmatrix} x'_{W_2} \\ y'_{W_2} \\ h_{O_2} \end{bmatrix} \\
\vec{P}_{W_3}^* & = \begin{bmatrix} x'_{W_3} \\ y'_{W_3} \\ h_{O_3} \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} \cos \theta_R & \sin \theta_R \cdot \sin \phi_R & \sin \theta_R \cdot \cos \phi_R \\ 0 & -\cos \phi_R & -\sin \phi_R \\ -\sin \theta_R & \cos \theta_R \cdot \sin \phi_R & \cos \theta_R \cdot \cos \phi_R \end{bmatrix} \cdot \begin{bmatrix} x_{W_3} \\ y_{W_3} \\ h_{O_3} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \cos \theta_R & \sin \theta_R \cdot \sin \phi_R & \sin \theta_R \cdot \cos \phi_R \\ 0 & -\cos \phi_R & -\sin \phi_R \\ -\sin \theta_R & \cos \theta_R \cdot \sin \phi_R & \cos \theta_R \cdot \cos \phi_R \end{bmatrix} \cdot \begin{bmatrix} x_{W_3} \\ y_{W_3} \\ h_{O_3} \end{bmatrix} = \begin{bmatrix} x'_{W_3} \\ y'_{W_3} \\ h_{O_3} \end{bmatrix} \\
\end{align*}
\]

(A.20)

A.4.2 Calculation of Robot Roll and Tilt

The trafficability of small obstacles and flat deepenings does not only depend on the ground clearance of a robot. These disturbances also cause the robot to roll and tilt which influences the sealing’s indentation (compare figure 3.9 in section 3.4.4). If the front wheel is positioned on a small step, the resulting robot tilt lets the frontal sealing expand whereas the rear part of the sealing is impressed.

After resolving these equations with the given height values $h_{O_2}$ and $h_{O_3}$ the angles for roll and tilt are known, as described by equation (A.23) and (A.24). The setup of this situation is given in figure A.2 with the position of the robot center $\vec{P}_{R|C}$ relative to the frontal wheel $W_1$. 

\[
\begin{align*}
\vec{P}_{W_2} & = \begin{bmatrix} x_{W_2} \\ y_{W_2} \\ h_{O_2} \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} \cos \theta_R & \sin \theta_R \cdot \sin \phi_R & \sin \theta_R \cdot \cos \phi_R \\ 0 & -\cos \phi_R & -\sin \phi_R \\ -\sin \theta_R & \cos \theta_R \cdot \sin \phi_R & \cos \theta_R \cdot \cos \phi_R \end{bmatrix} \cdot \begin{bmatrix} x_{W_2} \\ y_{W_2} \\ h_{O_2} \end{bmatrix} = \begin{bmatrix} x'_{W_2} \\ y'_{W_2} \\ h_{O_2} \end{bmatrix} \\
\end{align*}
\]

(A.21)
\[ \theta_R = \arcsin \left( \frac{h_{O_2} + h_{O_3}}{3 \cdot l_W} \right) \quad (A.23) \]
\[ \phi_R = \arcsin \left( \frac{h_{O_2} - h_{O_3}}{2 \cdot l_W \cdot \sin(60^\circ) \cdot \cos \theta_R} \right) \quad (A.24) \]

\textbf{Figure A.2:} Schematic view of the rolled and tilted wheel triangle with the normal vector \( \vec{n} \) of the rotated drive plane.

These two angles have to be transformed to angle \( \gamma_n \), which represents the total inclination of the spanned plane and angle \( \delta_n \) for the direction of it as shown in figure A.3. This description corresponds to the representation of a spherical coordinate system and is done via the following equation (A.25) and (A.26):

\[ \gamma_n = \arccos (\cos \theta_R \cdot \cos \phi_R) \]
\[ \delta_n = \begin{cases} \arctan \left( -\frac{\sin \phi}{\sin \theta_R \cdot \cos \phi_R} \right) & , \text{if } (\sin \theta_R \cdot \cos \phi_R) > 0 \\ \operatorname{sgn} (\sin \phi_R) & , \text{if } (\sin \theta_R \cdot \cos \phi_R) = 0 \\ \arctan \left( -\frac{\sin \phi}{\sin \theta_R \cdot \cos \phi_R} \right) + 180^\circ & , \text{if } (\sin \theta_R \cdot \cos \phi_R) < 0 \land \sin \phi_R \leq 0 \\ \arctan \left( -\frac{\sin \phi}{\sin \theta_R \cdot \cos \phi_R} \right) - 180^\circ & , \text{if } (\sin \theta_R \cdot \cos \phi_R) < 0 \land \sin \phi_R > 0 \end{cases} \quad (A.26) \]

The most interesting points are now the theoretical highest \( \vec{P}_{S|\text{high}} \) and the theoretical lowest point \( \vec{P}_{S|\text{low}} \) of the sealing circle, which correspond to the rotated sealing without
any adoptions by the air-filled tube. Based on these points, the desired indentation and expansion of the sealing, to keep the system air proof, can be calculated as shown below. At first, the center point $\vec{P}'_{R|C}$ of the robot relative to the front wheel is determined via equation (A.27), which performs the tilt and roll rotation of the original planar location $\vec{P}_{R|C} = (-l_W, 0.0, 0.0)^T$.

$$
\vec{P}'_{R|C} = R_z(0) \cdot R_y(\theta_R) \cdot R_x(\phi_R) \cdot \vec{P}_{R|C} = \begin{pmatrix}
-l_W \cdot \cos \theta_R \\
0 \\
l_W \cdot \sin \theta_R
\end{pmatrix}
$$

According to the spherical representation the lowest sealing point $\vec{P}_{S|low}$ uses an offset vector $\vec{V}_{S|offset}$ from the robot center, which is calculated as shown in equation (A.28). Figure A.4 illustrates the relationship between the different angles, points and heights.

$$
\vec{V}_{S|offset} = \begin{pmatrix}
-r_s \cdot \sin(\gamma_n + 90^\circ) \cdot \cos \delta_n \\
r_s \cdot \sin(\gamma_n + 90^\circ) \cdot \sin \delta_n \\
r_s \cdot \cos(\gamma_n + 90^\circ)
\end{pmatrix} = \begin{pmatrix}
r_s \cdot \cos \gamma_n \cdot \cos \delta_n \\
r_s \cdot \cos \gamma_n \cdot \sin \delta_n \\
-r_s \cdot \sin \gamma_n
\end{pmatrix}
$$

The resulting coordinate is the sum of the offset and the robot center as given in equation (A.29) whereas the coordinate of the highest sealing point uses equation (A.30).
\[ \vec{P}_{S|low} = \vec{P}_{R|C} + \vec{V}_S^{\text{offset}} \]  
(A.29)

\[ \vec{P}_{S|high} = \vec{P}_{R|C} - \vec{V}_S^{\text{offset}} \]  
(A.30)

Figure A.4: Calculation of the theoretical highest and lowest sealing points and the necessary indentation \( h_{S|\text{ind}} \) and expansion \( h_{S|\text{exp}} \) of the sealing to seal the system toward the ground plane.

Finally, the expansion of the sealing is assumed to be perpendicular to the plane given by the wheel triangle. Thus, the z-coordinates of the two points have to be taken and transformed to get the amount of needed indentation \( h_{S|\text{ind}} \) or expansion \( h_{S|\text{exp}} \) of the sealing tube according to equation (A.31) and (A.32) with the unit vector \( \vec{e}_z = (0, 0, 1) \).

Depending on the ground setup additional height offset values \( h_{S|\text{low/high}}^{\text{offset}} \) have to be added, e.g. if the robot faces a step instead of a single obstacle or if the frontal wheel is not located on the ground plane. These offset values are the height differences of the surface point of the frontal wheel and the ground points beyond the sealing.

\[ h_{S|\text{ind}} = \frac{\vec{e}_z \cdot \vec{P}_{S|low} - h_{S|low}^{\text{offset}}}{\cos \gamma_n} \]  
(A.31)

\[ h_{S|\text{exp}} = \frac{\vec{e}_z \cdot \vec{P}_{S|high} - h_{S|high}^{\text{offset}}}{\cos \gamma_n} \]  
(A.32)
A.4.3 Direct Calculation of Chamber Forces

Regarding the first precondition concerning the center chamber (compare section 7.4.4) it is possible to determine the force values for the chambers $C_1$, $C_3$ and $C_5$ according to figure 2.10. Equation (A.33) shows the three conditions which form the basis of this calculation.

\[
\begin{align*}
\frac{F_N}{2} &= F_{C_1} + F_{C_3} + F_{C_5} + \frac{F_{C_7}}{2} \quad = 0 \\
x_F \cdot \frac{F_N}{2} &= x_{C_1} \cdot F_{C_1} + x_{C_3} \cdot F_{C_3} + x_{C_5} \cdot F_{C_5} + x_{C_7} \cdot \frac{F_{C_7}}{2} \\
y_F \cdot \frac{F_N}{2} &= y_{C_1} \cdot F_{C_1} + y_{C_3} \cdot F_{C_3} + y_{C_5} \cdot F_{C_5} + y_{C_7} \cdot \frac{F_{C_7}}{2} \quad = 0
\end{align*}
\]  

(A.33)

Here additionally the force of the center chamber $F_{C_7}$ has to be considered because it already generates a portion of downforce which works contrarily to the other chambers, if the desired pressure point lies outside of the robot center. Because of the geometric setup the values $y_{C_1}$, $x_{C_7}$ and $y_{C_7}$ are zero in robot coordinates and can be neglected. This leads to the forces of the three considered chambers as shown in equation (A.34) depending on desired downforce, force point and geometric setup:

\[
\begin{align*}
F_{C_5} &= \frac{(y_F \cdot x_{C_1} - y_F \cdot x_{C_3} - y_{C_3} \cdot x_{C_1} + y_{C_3} \cdot x_F) \cdot \frac{F_N}{2} + x_{C_1} \cdot y_{C_3} \cdot \frac{F_{C_7}}{2}}{x_{C_1} - x_{C_3}} \\
F_{C_3} &= \frac{(x_F - x_{C_1}) \cdot \frac{F_N}{2} - x_{C_1} \cdot F_{C_3} + x_{C_5} \cdot F_{C_5} - x_{C_1} \cdot \frac{F_{C_7}}{2}}{x_{C_1} - x_{C_3}} \\
F_{C_1} &= \frac{x_F \cdot \frac{F_N}{2} - x_{C_3} \cdot F_{C_3} - x_{C_5} \cdot F_{C_5}}{x_{C_1}}
\end{align*}
\]  

(A.34)

The calculations of the downforce point based on the preconditions in equation (A.33) are correct because of the geometric setup of the seven chambers (compare figure 2.10). By using the symmetrical setup of the negative pressure system and equation (2.10) for the determination of the chamber center points it is possible to proof the correctness of the calculations. For clearness, only the chambers $C_1$, $C_3$ and $C_5$ will be regarded, but this explanation is identical for the other three outer chambers $C_2$, $C_4$ and $C_6$. The symmetrical chamber setup leads to a balanced downforce position, if all three chamber downforces are equal. In this case, the combined downforce point lies in the robot center whereas it lies in the center point of an adhered chamber, if the other chamber forces are zero.

The same calculations concerning the desired chamber forces can be done for the other three outer working chambers 2, 4 and 6. The basic equation (A.35) for this chamber setup is similar to the previous equation (A.33).
\[
\begin{align*}
\frac{F_N}{2} &= F_{C_2} + F_{C_4} + F_{C_6} + \frac{F_{C_7}}{2} \\
x_{F_N} \cdot \frac{F_N}{2} &= x_{C_2} \cdot F_{C_2} + x_{C_4} \cdot F_{C_4} + x_{C_6} \cdot F_{C_6} + x_{C_7} \cdot \frac{F_{C_7}}{2} \\
y_{F_N} \cdot \frac{F_N}{2} &= y_{C_2} \cdot F_{C_2} + y_{C_4} \cdot F_{C_4} + y_{C_6} \cdot F_{C_6} + y_{C_7} \cdot \frac{F_{C_7}}{2}
\end{align*}
\]

(A.35)

Similar to the previous consideration some terms can be left out because they are zero: \(y_{C_4}, x_{C_7}\) and \(y_{C_7}\). This equation leads to the remaining forces of chambers \(C_2, C_4\) and \(C_6\) as shown in equation (A.36):

\[
\begin{align*}
F_{C_6} &= \frac{(y_{F_N} \cdot x_{C_4} - y_{F_N} \cdot x_{C_2} - y_{C_2} \cdot x_{C_4} + y_{C_2} \cdot x_{F_N}) \cdot \frac{F_N}{2} + x_{C_4} \cdot y_{C_2} \cdot \frac{F_{C_7}}{2}}{y_{C_6} \cdot x_{C_4} - y_{C_6} \cdot x_{C_2} - y_{C_2} \cdot x_{C_4} + y_{C_2} \cdot x_{C_6}} \\
F_{C_2} &= \frac{(x_{C_4} - x_{F_N}) \cdot \frac{F_N}{2} - x_{C_4} \cdot F_{C_6} + x_{C_6} \cdot F_{C_6} - x_{C_4} \cdot \frac{F_{C_7}}{2}}{x_{C_4} - x_{C_2}} \\
F_{C_4} &= \frac{x_{F_N} \cdot \frac{F_N}{2} - x_{C_2} \cdot F_{C_2} - x_{C_6} \cdot F_{C_6}}{x_{C_4}}
\end{align*}
\]

(A.36)
A.5 Safeguarding and Safety Analysis

This section describes a guideline for safeguarding processes used by several international standards and some analytic methods for the identification of hazards and causal relationships.

A.5.1 Standardized Safeguarding Process

The international standard IEC 61508-1 formulates design guidelines to achieve an appropriate safety integrity level (SIL) in terms of a standardized safeguarding process [IEC61508]. Figure A.5 shows how predictable risks should be handled within the development process according to the international standards ISO 12100 and ISO 14121, which are used by IEC 61508-1 [ISO12100, ISO14121].

![Diagram of Development Scheme](image)

**Figure A.5:** Development scheme to handle predictable risks according to ISO 14121: risk analysis, risk rating and system optimization.

**Risk assessment** First of all, a risk analysis has to be performed using one or more examination techniques. This includes a specification of the limits of the system, the identification of possible hazards and an estimation of the risks. After this the identified risks are rated, which may also include the determination of the safety
integrity level via *risk graphs* or a layer-of-protection analysis [Gulland2004]. Both analysis and rating steps are known as *risk assessment* and are dependend on the robotic system, its application and the robot environment. Finally it has to be determined if the current risk is tolerable or not.

**Safety measures by the engineer** The next step are *safety measures*, which are applied by the construction engineer. This includes e.g. risk reduction techniques like an intrinsic safe mechanical construction or technical measures in terms of special controllers [Radandt2007]. If the current risks are not tolerable, the system has to be adapted and checked for new risks. The implementation of these safety methods in the range of industrial robots is more or less straight forward: Beside mechanical stoppers, e.g. to limit joint angles to the desired range, most of these methods are realized via controllers or electronic circuits. Emergency stops will be performed by relais circuits, which can shut down the power supply of the robot, another option is a halt of the robot at the current position, which has to be done by the joint controllers. The limitation of power, force and velocity conform to the international standard ISO 10218-1 as described before is more complex but can also be done by special controllers. Additional external measures can be a safety fence or a safety light curtain. In every case the preparation of important user information like warning signals at the machine or safety instructions in the manual are further important measures.

**Safety measures by the user** The last step is the realization of safety measures, which address the operator of the system. This includes the compliance of the safety instructions, the usage of personal protective equipment, the execution of regular inspections of the system and of employee trainings.

### A.5.2 Overview of Standard Analysis Methods

In literature, a large amount of different analysis methods can be found. For further techniques and a more precise description, the publication by Wilson and McDermid [Wilson1995] as well as the book of Dhillon might be useful [Dhillon2005]. A large collection of analysis and assessment methods has also been published by the government of the United Kingdom [DCLC2008].

**Cause and effect diagram (CAED)** This method has been originally developed for quality control, but can be used for safety analysis as well [Mears1995]. The cause and effect diagram, which is also known as *fishbone diagram* (see figure A.6) is divided into two parts: on the right side a possible effect is noticed, left of it all possible causes are given. This diagram can be applied to identify safety-related problems in the range of mobile robots. However, it allows only a causal statement without considering additional knowledge like e.g. probabilities of the individual causes or severity classes.

**Failure mode and effect analysis (FMEA)** Failure mode and effect analysis is similar to CAED and lists potential failure modes of all components and their effects on the listed elements and the environment [Jordan1972]. Based on collected data, the probabilities of the failure modes are determined and critical modes can be reviewed.
A.5. Safeguarding and Safety Analysis

In general, it is used to show that no single point of failure can lead to a critical event and to attest that safety goals for a system have been achieved.

Technic of operations review (TOR) This hands-on method has been developed to determine the cause of an accident or an operation malfunction. It uses general system knowledge as well as the experiences of the human workers and makes use of a sheet containing terms with simple yes/no decisions. Unfortunately, TOR is designed to be used after-the-fact [Dhillon2003].

Nuclear safety cross-check analysis (NSCCA) This software analysis technique is used to determine a confidence value, that the control software of a technical system will not cause a malfunction. This method was originally developed to certify nuclear weapon software and firmware of the United States Air Force [Dhillon2003]. The idea is to break down the software into simplest functions, which can be reviewed and verified. The combination of these atomic functions is then checked via security and control measures.

Interface safety analysis (ISA) Another source of safety problems are incompatibilities between different subsystems. The interface safety analysis addresses this problem by taking different kinds of interfaces into account under the precondition that each individual system is safe. The relationships can be split into the three categories functional, flow and physical depending on the type of items, which can be electronic components but also devices like water pumps or air valves [Hammer1993].

Probability tree analysis This analysis method is used to determine the reliability of human actions in cooperation with machines or robots by generating a probability tree. In this case the likelihood of success or failure of each critical action of a worker is identified, the total chance of success for a given situation can then be obtained by summing up the corresponding probabilities. Also factors like stress or equipment failures can be considered by this method.

Markov method The mathematical markov analysis is widely used to receive reliability values of engineering systems. The preconditions of this method are constant or time independent occurrence rates of failures [Dhillon2005]. Via the markov method it

![Figure A.6: Example for a cause and effect diagram for a sudden halt of a robotic arm. Possible causes for an effect are connect via the center line.](image-url)
is possible for instance to calculate the probabilities of the system to enter either a safe or an unsafe failure state. Similar to other analysis methods the foundation of this approach is an explicit identification of the system states as well as failure rates.

**Dynamic event tree analysis method (DETAM)** This method has been developed for the analysis of dynamic systems and allows a time-dependent processing of system states [Acosta1993]. DETAM uses an *event tree*, which allows branching at different times. Via system variables the current system state and possible branches are determined, while branching rules decide when a branching should take place. It can be applied to describe operator behaviors, model the consequences, and to analyze the causal sources of errors. In fact, this approach seems to be an outstanding method to analyze a dynamic system like a climbing robot. Nevertheless, a detailed description of the dynamic elements and effects is needed for this analysis technique.

**Risk graph method** Risk graphs are widely used to determine the safety integrity level (SIL) of a safety-related system and the demand of additional risk reduction techniques – like in the international standard IEC 61508-5 [IEC61508]. The SIL is a measurement unit of the quality of a system and can be used as an indicator of the expected performance of the system. In contrast to common literature the four levels as shown in table A.5 have to be determined depending on the frequency of the safety function [Gulland2004]. For functions with a low rate of demand (e.g. anti-lock braking or air bags), the probability of a system fail on demand (PFD) is important whereas continuous functions like steering and braking use the mean time to failure (MTTF).

<table>
<thead>
<tr>
<th>SIL</th>
<th>range of PFD</th>
<th>range of MTTF (years)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$10^{-9} \leq \text{PFD} &lt; 10^{-4}$</td>
<td>$10^5 \geq \text{MTTF} &gt; 10^4$</td>
</tr>
<tr>
<td>3</td>
<td>$10^{-4} \leq \text{PFD} &lt; 10^{-3}$</td>
<td>$10^4 \geq \text{MTTF} &gt; 10^3$</td>
</tr>
<tr>
<td>2</td>
<td>$10^{-3} \leq \text{PFD} &lt; 10^{-2}$</td>
<td>$10^3 \geq \text{MTTF} &gt; 10^2$</td>
</tr>
<tr>
<td>1</td>
<td>$10^{-2} \leq \text{PFD} &lt; 10^{-1}$</td>
<td>$10^2 \geq \text{MTTF} &gt; 10^1$</td>
</tr>
</tbody>
</table>

The risk graph method applies four risk parameters, which can be qualitative (as given here) but also quantitative descriptions:

- **Consequence (C):** Consequence of the hazard for a person, which is divided into four categories: Minor injury ($C_1$); serious permanent injury of one or more persons, death to one person ($C_2$); death to several persons ($C_3$) and death to many persons ($C_4$).
- **Exposure (F):** Two classes are defined for the frequency of the hazardous zone and its exposure time: Rare to more often exposure ($F_1$) and frequent to permanent exposure in the hazardous zone ($F_2$).
- **Avoidability (P):** The possibility to avoid the hazard can either be possible under certain conditions ($P_1$) or almost impossible ($P_2$).
- **Demand rate** ($W$): This factor estimates the frequency of the unwanted occurrence without avoidance measures and can be described as the probability of the hazard to occur. In this case it can be a very slight probability with only a few unwanted occurrences ($W_1$); a slight probability with few occurrences ($W_2$) or a relatively high probability with frequent occurrences ($W_3$).

![Risk Graph](image)

**Figure A.7**: A typical risk graph as used in many risk analysis.

A typical risk graph is shown in figure A.7 with the different decision steps and the result depending on the demand rate in the box on the right side. In the case of # no safety requirements exist, $a$ indicates no special requirements whereas $1-4$ are the four SIL. In the worst case $b$ a single safety system is not sufficient.
A.6 Modular Control Architecture MCA2-KL

The software framework MCA2 – initially created by the Forschungzentrum Informatik (FZI) in Karlsruhe and developed further by the University of Kaiserslautern\(^1\) – has been chosen for the implementation of the control software of CROMSCI because of its modular structure. This architecture supports the creation of large and complex control elements based on single components, the so called modules as shown in figure A.8.

A.6.1 Software Components

The main benefit of this architecture is its modularity. The control software consists of single modules, which exchange control and sensor data. Therefore, it is possible to reuse these modules or larger control groups for other projects, which makes it possible to share functional software elements between different robots. This gives researchers a common basis of fundamental functions, controllers or strategies.

**Module** A module is the basic software component providing several in- and outputs as depicted in figure A.8a. On the right side (red), a module receives control data in terms of float values of user-defined dimension. On the left side (yellow), it receives so-called sensor data, which can be raw or filtered values from a sensor but also abstract (or virtual) sensor data from other software modules. Based on its internal function, a module calculates control values and/or virtual sensor data, which are given to the corresponding outputs.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{mca2_module.png}
\caption{(a) Basic software module.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{mca2_group.png}
\caption{(b) Group of software modules.}
\end{figure}

**Figure A.8:** The basic structural components of the MCA2 framework: Software modules with different in- and outputs (a) and a group of modules (b) which acts like a single module.

\(^{1}\)http://rrlib.cs.uni-kl.de/
Group To create a hierarchical structure and to organize associated elements, two or more of these modules can be combined into a group (figure A.8b). Observed from the outside, a group acts like a single module with controller and sensor in- and outputs. Of course, the group has to forward its inputs to the internal modules. In the same way it has to collect those outputs of the modules which should be provided as group outputs.

Part A part is the executional program which consist of one MCA2 group. Inside of this group any kind of hierarchical structure can be realized. Similar to modules and groups, also a part provides an interface for the exchange of sensor and control informations with other parts.

Edge An edge – depicted in figure A.8b as an arrow – is the basic interface for the exchange of data. It supports an arbitrary number of float values which can be send from one module, group or part to another.

Blackboard More complex data like camera images or distance informations from a laser range sensor cannot be transmitted via edges. Therefore, a data structure called blackboard has been implemented. A blackboard can contain any kind of user- or predefined data, which can also be resized during runtime. Various access methods allow an easy handling of the internal data.

So far, all robots of the Robotics Research Lab have been implemented based on MCA2. Nevertheless, there is an architectural switch to the more sophisticated architecture FINROC\textsuperscript{2} which handles some drawbacks of MCA2 and other common control architectures [Reichardt2012].

A.6.2 MCAbrowser

The MCAbrowser is an important tool during the development phase of the control software, but also during runtime. As shown in figure A.9, it can be used to visualize the internal structure of the single control modules including their hierarchy and connections. In the normal view (top left), the individual groups and modules are shown, but for the analysis of a behavior-based network it is possible to switch in the iB2C mode (right) which visualizes the current behavior states (activity, activation and target rating; compare section 2.2.1) as well as their current mutual influence. Especially this aspect makes the tool very helpful to find errors or invalid states inside of such a network, as already depicted in figure 7.22.

Furthermore, it is possible to take a look at the current values (bottom left), which are exchanged by the software modules. This feature supports debugging of the software tremendously to find erroneous or unexpected values inside of the system. It is also possible to change user-defined parameters of the software modules online. This allows e.g. a dedicated tuning of control parameters during runtime without any restart of the software.

\textsuperscript{2}http://www.finroc.org/
A.6.3 **MCAgui**

The second tool related to MCA2 is the MCAgui, which allows an easy creation of a customized user interface as shown in figure A.10 (top). It behaves similar to a MCA2 software module: Each element can be connected to one or more input or output interfaces of the running software parts, either to control the software (e.g. via buttons and sliders) or to visualize informations about its state via bars, graphs or text widgets. Here, the GUI is used to control the locomotion of the robot and to set the desired values of the adhesion behaviors and the risk prediction components.

Figure A.10 (bottom) shows the GUI used to visualize the optimization process of the training phase. In the main window on the right side, the developing of the adhesion rating values of the selected training set (black). In red the current evaluation values are drawn based on the best set of weights (individual), which could be found so far. Anomalies causing a mallus because of a missing or an unwanted value are highlighted via bars to visualize further needs of optimization.

---

**Figure A.9**: Screenshot of the MCAbrowser.
A.6. Modular Control Architecture

logging and test replay
behavior control and meta data
developing of values for debugging
control panel
highlightning of undesired or missing values

Figure A.10: Screenshot of the main window of the graphical user interface for CROMSCI (top) and of the window showing the offline optimization process (bottom).
A.7 Implementation Details

The complete software of CROMSCI has been realized using MCA2. Figure A.11 shows the three parts of the software: a **control part** performing the higher control including kinematic calculations and motion behaviors, a **realtime part**\(^3\) which is responsible for the closed-loop adhesion control (compare chapter 7), the risk prediction function (see chapter 8) and the connection to the robot hardware, and the **simulation part** below, if it is needed.

As it can be seen, there exists a hardware abstraction layer module forwarding the in-and outputs of the subjacent elements. But, it provides a general interface for the control structure above to access hardware and simulation in the same way. By this mechanism it is possible to generate either the simulation part, the hardware access, or a mixture of it, without changing the higher control components.

\(^3\)The realtime part runs with a short cycle time of about 10ms, but does not fulfill hard realtime conditions since the system does not run on RT Linux.
A.7. Implementation Details

A.7.1 Control Software

The elements of the control part are illustrated in figure A.12. These groups are responsible for a couple of tasks related to robot navigation and operation and contain a couple of further modules.

Figure A.12: Structure of the control part, which is responsible for sensing, mapping and control of the robot. BB and GBB denote the usage of (geometry) blackboards.

- **Kinematic** This group performs the kinematic calculations (forward and backward) and adaptations of the wheel control as described in sections 2.1.2 and 6.1.

- **Localization** So far, only odometry calculations are implemented here according to section 2.1.2. A second external localization method based on a fix laser scanner attached to the building has also been realized here, but is not used by default.

- **Filtering** This group executes some filtering operations and transformations of sensor data. The laser ranger data e.g. are transformed here from the sensor’s polar coordinates into cartesian robot coordinates.
**Mapping** The mapping group includes the calculation of the sector map (section 9.1.1) as well as a gridmap shown in figure A.13. This map will be used for further applications, if a reliable localization method exists.

![Gridmap](image)

**Figure A.13:** Visualization of the gridmap showing different characteristics like estimated leakage, pressure trend or maximal downforce.

**Behaviors** Here, the motion and evasion behaviors for drive and manipulator arm are located according to the descriptions in sections 2.2.2, 9.1.2 and 9.2.3.

**Visualization** For operation and debugging of the robot, this group prepares the sensor and state values so that they are better readable than simple numeric values. This includes e.g. the creation of so-called geometry blackboards (GBB) which draw geometric primitives as shown in figure A.10 in the top left corner.

All these components work in the same manner if the real hardware or just the simulation is used. Differences between simulation and the real robot are handled via parameters, which are loaded on startup depending on the used hardware components.

### A.7.2 Simulation

For simulation purposes, mca2 uses its own library SimVis3D\(^4\) developed at the University of Kaiserslautern [Braun2007]. This tool allows it to create complex scenes based on three-dimensional models and an XML description which combines the single elements. Listing 1 shows the definition of the simulated scene. The first part contains the surrounding environment for decoration purposes in OpenInventor file `outdoor_scene_no_bridge.iv` (line 4). This environment is the basic element, all other components are relative to its coordinates. This counts also for the robot mounting point (line 8), which denotes the pose of the robot torso, manipulator parts and its drive components.

\(^4\)http://rrlib.cs.uni-kl.de/software/simvis3d/
A.7. Implementation Details

The simulated bridge consists of single cubes with a edge length of 2 m. The advantage of this setup is the easy exchange of cubes via the description file as shown in the second part of listing A.7.2. Since the cubes can have a structured surface, cracks, or holes as well as large obstacles on them, it is possible to change the appearance of the bridge by a simple replacement of cubes or to add new cubes. This cube-based structure can be seen in figure A.14.

Listing 1: Description of simulation components: Robot torso and movable parts.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<content unit="meters">
<!-- >>>>>>>>>>>>>>>>> ENVIRONMENT <<<<<<<<<<<<<< -->
<part file="cromsci-vis-obj/outdoor_environment/outdoor_scene_no_bridge.iv" name="ENVIRONMENT" attached_to="ROOT" pose_offset="0 0 0 0 0 0"/>

<!-- >>>>>>>>>>>>>>>>> CROMSCI <<<<<<<<<<<<<< -->
<!-- ----------- robot torso ----------- -->
<part file="cromsci-vis-obj/empty_node.iv" name="CROMSCI_MOUNT" attached_to="ENVIRONMENT" pose_offset="-1 -8 7 0 -90 0"/>

<!-- ----------- manipulator ----------- -->
<part file="cromsci-vis-obj/manip_schlitten-links.iv" name="SLEDGE_LEFT" attached_to="CROMSCI_TORSO" pose_offset="0 0 0.18 0 0 0"/>
<part file="cromsci-vis-obj/manip_schlitten-rechts.iv" name="SLEDGE_RIGHT" attached_to="CROMSCI_TORSO" pose_offset="0 0 0.18 0 0 0"/>

<!-- ----------- drive_front ----------- -->
<part file="cromsci-vis-obj/antrieb-dreheinheit.iv" name="DRIVE_FRONT" attached_to="CROMSCI_TORSO" pose_offset="0 0 0 0 0 0"/>
<part file="cromsci-vis-obj/antrieb-dreheinheit.iv" name="DRIVE_LEFT" attached_to="CROMSCI_TORSO" pose_offset="0 0 0 0 0 0"/>

<!-- ----------- drive_right ----------- -->
<part file="cromsci-vis-obj/antrieb-dreheinheit.iv" name="DRIVE_RIGHT" attached_to="CROMSCI_TORSO" pose_offset="0 0 0 0 0 0"/>
```
Listing 1 (continued): Description of simulation components: Single bridge cubes.

```xml
<!-- ----------- 1. row ----------- -->
<part file="cromsci/vis_obj/empty_node.iv" name="BRIDGE_MOUNT"
      attach_to="ROOT" pose_offset="0 0 1 90 0 270" />

<!-- ----------- 2. row ----------- -->
<part file="cromsci/vis_obj/empty_node.iv" name="BRIDGE_MOUNT"
      attach_to="ROOT" pose_offset="0 0 1 90 0 270" />

<!-- ----------- 3. row ----------- -->
<part file="cromsci/vis_obj/bridge/cube_crack_02_2x2x2m_2cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="0 4 0 0 0 270" />
<part file="cromsci/vis_obj/bridge/cube_crack_02_2x2x2m_2cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="2 4 0 0 0 0" />
<part file="cromsci/vis_obj/bridge/cube_crack_01_2x2x2m_2cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="4 4 0 0 0 0" />
<part file="cromsci/vis_obj/bridge/cube_crack_01_2x2x2m_2cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="6 4 0 0 0 0" />
<part file="cromsci/vis_obj/bridge/cube_crack_01_2x2x2m_2cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="8 4 0 0 0 270" />
<part file="cromsci/vis_obj/bridge/cube_crack_01_2x2x2m_2cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="10 4 0 0 0 0" />

<!-- ----------- 4. row ----------- -->
<part file="cromsci/vis_obj/bridge/cube_crack_01_2x2x2m_20cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="0 6 0 0 0 0" />
<part file="cromsci/vis_obj/bridge/cube_crack_01_2x2x2m_20cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="2 6 0 0 0 0" />
<part file="cromsci/vis_obj/bridge/cube_crack_01_2x2x2m_20cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="4 6 0 0 0 0" />
<part file="cromsci/vis_obj/bridge/cube_crack_01_2x2x2m_20cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="6 6 0 0 0 0" />
<part file="cromsci/vis_obj/bridge/cube_crack_01_2x2x2m_20cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="8 6 0 0 0 0" />
<part file="cromsci/vis_obj/bridge/cube_crack_01_2x2x2m_20cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="10 6 0 0 0 0" />

<!-- ----------- 5. row ----------- -->
<part file="cromsci/vis_obj/bridge/cube_structure_01_2x2x2m_20cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="0 8 0 0 0 180" />
<part file="cromsci/vis_obj/bridge/cube_structure_01_2x2x2m_20cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="2 8 0 0 0 90" />
<part file="cromsci/vis_obj/bridge/cube_structure_01_2x2x2m_20cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="4 8 0 0 0 0" />
<part file="cromsci/vis_obj/bridge/cube_structure_01_2x2x2m_20cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="6 8 0 0 0 0" />
<part file="cromsci/vis_obj/bridge/cube_structure_01_2x2x2m_20cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="8 8 0 0 0 90" />
<part file="cromsci/vis_obj/bridge/cube_crack_08_2x2x2m_2cm.iv" name="CUBE"
      attach_to="BRIDGE_MOUNT" pose_offset="10 8 0 0 0 90" />

<!-- ----------- bridge top ----------- -->
<part file="cromsci/vis_obj/bridge/bridge_top.iv" name="BRIDGE_TOP"
      attach_to="BRIDGE_MOUNT" pose_offset="1 18 0 270 0 0" />
```

Listing 1 (continued): Description of simulation components: Simulated sensors.

```xml
<!-- ----------- depth camera ----------- -->
<camera name="rob_cam" type="perspective" vfov="45" near_limit="0.1"
        attach_to="DEPTH_CAMERA" />
<distance_sensor name="lms200" max_distance="4" scan_angle_range="240" angular_resolution="0.352433" sensor_offset="0 0 0.35 0 20 0" image_width="640" image_height="100" near_distance="0.05"
                 far_distance="4." scan_mode="depth_buffer" attach_to="TCP" />
```
Beside the visual point of view, it is also possible to append different kind of simulated sensors to the scene, e.g. cameras or laser range sensors. Again, this is done inside of the description file. The third part of listing A.7.2 contains the depth camera (line 14) used for sealing simulation to measure the ground height and the laser scanner (line 35) for obstacle avoidance. It can be seen, that the simulated laser range sensor is attached to the TCP and not to the robot itself, so it is moved in the same way as the manipulator head.

Figure A.14: Wireframe view of a simulated bridge pylon.

Figure A.15: Simulation of the robot environment and visualization via SimVis3D (left) and extracted depth image (right) used for sealing simulation.
Figure A.15 shows the rendered view of the simulation (left side) with the robot attached to a wall with structured surface and a deep crack below it. Based on the depth information of the surface below the robot and a simulated elastic characteristics of the air-filled sealing, the chamber leakages of the corresponding areas are calculated, as visualized on the right side in figure A.15. Additionally, the SimVis3D framework is used to simulate the laser range sensor for testing the detection and evasion of obstacles (compare section 9.1).

Since the SimVis3D framework does not support physical simulation by default, it has to be extended via additional simulators. In the present case the NVIDIA PhysX engine\(^5\) has been chosen since it is powerful enough to calculate realistic robot behavior but also very fast due to an integrated support by actual graphic processors from NVIDIA. Even without dedicated support from the graphics card, this physics engine is fast enough to run additionally to the remaining mca2 simulation and control software. Figure A.16 shows a screenshot of the visualization of the physical simulation. Here, the robot is pressed with a certain downforce (adhesion force) to the ground. A sideward pulling force simulates the gravity and allows it to simulate a robot drop-off.

![Image of physical simulation based on the NVIDIA PhysX engine](http://www.nvidia.com/object/physx_faq.html)

Figure A.16: Physical simulation based on the NVIDIA PhysX engine

Beside the SimVis3D framework and PhysX engine additional simulation components exist to get a realistic behavior from the robot hardware like the drive units, manipulator components, pressure valves and vacuum chambers. Figure A.17 shows the single simulation components, which interact with each other via edges (arrows) and blackboards (BB, dashed arrows). The central element are the SimVis3D components in terms of the Coin\(^6\)

---

graphics library which contains the 3d scene and a laser scanner module simulating a laser range sensor based on the scene information. Inside of the same group also the leakage simulation is located, which calculates leakage values based on the surface structure below the sealing and an approximated elasticity characteristics.

Valve positions and leakage information are given to the thermodynamic model to calculate the air flow and chamber pressures [Wettach2005a]. The PhysX engine receives simulated drive velocities and positions, and influences the robot model (e.g. via adhesion force, gravity) for a realistic motion behavior. The final position is given to the scene simulation. One big advance of this simulation network is its modularity which makes it possible to replace single hardware elements by the corresponding simulation component or vice versa. By this mechanism it is e.g. possible to test the operability of the real chamber valves with simulated leakages, airflows and pressure values.

Figure A.17: Structure of the simulation components.
A.8 Behavior Values and Acquired Weights for Risk Prediction

The evaluation function $F_E(\vec{a}, \vec{r})$ from section 8.1.2 depends on the weights (optimized via the genetic algorithm) and meta values of the behaviors inside of the adhesion control network. Listings 2 shows the header of the training sets and the number of behavior values used for risk prediction. In the first line, information about the number of behavior values (here: 90), the rating values (here: 3), and the cycle time (here: 10 ms) are given. The second line contains a list of names, each of them consisting of five components as in the example of $[1]_{SI\_Activity\_Fusion\_Force\_Control\_Value}$:

- Index number of value (here: 1).
- Type of edge like sensor input, controller output, etc. (here: SI for sensor input).
- Meta value, which can be activity, target rating, or activation (here: Activity).
- Name of the behavior or value (here: Fusion_Force_Control).
- Type of value (here: Value, which is the raw one). This can be also a low pass filtered value (smoothed), median, average, or further derivations.

The first 90 entries in this list are behavior inputs used for evaluation, the last three names denote – according to the first header line – those three values, which might be used as values for comparison and optimization by the rating function. In this case, these are the adhesion score values introduced in section 8.2.1.

Listing 2: Header lines of a training set showing the involved and evaluated behavior values.

```bash
# evaluation 90 3 10
# [1]_SI_Activity_Fusion_Force_Control_Value
[2]_SI_Activity_Fusion_Force_Control_Value_Smoothed
[3]_SI_Target_Rating_Fusion_Force_Control_Value
[4]_SI_Target_Rating_Fusion_Force_Control_Value_Smoothed
[5]_SI_Activity_Force_Value_Control_Value
[6]_SI_Activity_Force_Value_Control_Value_Smoothed
[7]_SI_Target_Rating_Force_Value_Control_Value
[8]_SI_Target_Rating_Force_Value_Control_Value_Smoothed
[9]_SI_Activity_Force_Point_Control_Value
[10]_SI_Activity_Force_Point_Control_Value_Smoothed
[11]_SI_Target_Rating_Force_Point_Control_Value
[12]_SI_Target_Rating_Force_Point_Control_Value_Smoothed
[13]_SI_Activity_Chamber_Pressure_Calculator_Value
[14]_SI_Activity_Chamber_Pressure_Calculator_Value_Smoothed
[15]_SI_Target_Rating_Chamber_Pressure_Calculator_Value
[16]_SI_Target_Rating_Chamber_Pressure_Calculator_Value_Smoothed
[17]_SI_Activity_Chamber_Deactivator_Value
[18]_SI_Activity_Chamber_Deactivator_Value_Smoothed
[19]_SI_Target_Rating_Chamber_Deactivator_Value
[20]_SI_Target_Rating_Chamber_Deactivator_Value_Smoothed
[21]_SI_Activity_Reservoir_Control_Value
[22]_SI_Activity_Reservoir_Control_Value_Smoothed
[23]_SI_Target_Rating_Reservoir_Control_Value
[24]_SI_Target_Rating_Reservoir_Control_Value_Smoothed
[25]_SI_Activity_Fusion_Chamber_Control_Value
[26]_SI_Activity_Fusion_Chamber_Control_Value_Smoothed
[27]_SI_Target_Rating_Fusion_Chamber_Control_Value
[28]_SI_Target_Rating_Fusion_Chamber_Control_Value_Smoothed
[29]_SI_Activity_Chamber_Control_Group_Value
```
Listing 2 (continued): Header lines of a training set.

[30] SI_Activity_Chamber_Control_Group_Value_Smoothed
[31] SI_Target_Rating_Chamber_Control_Group_Value
[32] SI_Target_Rating_Chamber_Control_Group_Value_Smoothed
[33] SI_Activity_Chamber_Control_Top_Value
[34] SI_Target_Rating_Chamber_Control_Top_Value_Smoothed
[35] SI_Activity_Chamber_Control_Top_Value
[36] SI_Target_Rating_Chamber_Control_Top_Value_Smoothed
[37] SI_Activity_Chamber_Control_Topleft_Value
[38] SI_Target_Rating_Chamber_Control_Topleft_Value_Smoothed
[39] SI_Activity_Chamber_Control_Topleft_Value
[40] SI_Target_Rating_Chamber_Control_Topleft_Value_Smoothed
[41] SI_Activity_Chamber_Control_Bottomleft_Value
[42] SI_Target_Rating_Chamber_Control_Bottomleft_Value_Smoothed
[43] SI_Activity_Chamber_Control_Bottomleft_Value
[44] SI_Target_Rating_Chamber_Control_Bottomleft_Value_Smoothed
[45] SI_Activity_Chamber_Control_Bottom_Value
[46] SI_Target_Rating_Chamber_Control_Bottom_Value_Smoothed
[47] SI_Activity_Chamber_Control_Bottom_Value
[48] SI_Target_Rating_Chamber_Control_Bottom_Value_Smoothed
[49] SI_Activity_Chamber_Control_Bottomright_Value
[50] SI_Target_Rating_Chamber_Control_Bottomright_Value_Smoothed
[51] SI_Activity_Chamber_Control_Bottomright_Value
[52] SI_Target_Rating_Chamber_Control_Bottomright_Value_Smoothed
[53] SI_Activity_Chamber_Control_Topright_Value
[54] SI_Target_Rating_Chamber_Control_Topright_Value_Smoothed
[55] SI_Activity_Chamber_Control_Topright_Value
[56] SI_Target_Rating_Chamber_Control_Topright_Value_Smoothed
[57] SI_Activity_Chamber_Control_Center_Value
[58] SI_Target_Rating_Chamber_Control_Center_Value_Smoothed
[59] SI_Activity_Chamber_Control_Center_Value
[60] SI_Target_Rating_Chamber_Control_Center_Value_Smoothed
[61] SI_Activity_Chamber_Control_Pressure_Trend_Value
[62] SI_Target_Rating_Chamber_Control_Pressure_Trend_Value_Smoothed
[63] SI_Activity_Chamber_Control_Pressure_Trend_Top_Value
[64] SI_Target_Rating_Chamber_Control_Pressure_Trend_Top_Value_Smoothed
[65] SI_Activity_Chamber_Control_Pressure_Trend_Topleft_Value
[66] SI_Target_Rating_Chamber_Control_Pressure_Trend_Topleft_Value_Smoothed
[67] SI_Activity_Chamber_Control_Pressure_Trend_Bottomleft_Value
[68] SI_Target_Rating_Chamber_Control_Pressure_Trend_Bottomleft_Value_Smoothed
[69] SI_Activity_Chamber_Control_Pressure_Trend_Bottom_Value
[70] SI_Target_Rating_Chamber_Control_Pressure_Trend_Bottom_Value_Smoothed
[71] SI_Activity_Chamber_Control_Pressure_Trend_Bottomright_Value
[72] SI_Target_Rating_Chamber_Control_Pressure_Trend_Bottomright_Value_Smoothed
[73] SI_Activity_Chamber_Control_Pressure_Trend_Topright_Value
[74] SI_Target_Rating_Chamber_Control_Pressure_Trend_Topright_Value_Smoothed
[75] SI_Activity_Chamber_Control_Pressure_Trend_Center_Value
[76] SI_Target_Rating_Chamber_Control_Pressure_Trend_Center_Value_Smoothed
[77] SI_Activity_Chamber_Control_Pressure_Trend_Center_Value
[78] SI_Target_Rating_Chamber_Control_Pressure_Trend_Center_Value_Smoothed
[79] SI_Target_Rating_Chamber_Leakage_Estimation_Top_Value
[80] SI_Target_Rating_Chamber_Leakage_Estimation_Top_Value_Smoothed
[81] SI_Target_Rating_Chamber_Leakage_Estimation_Topleft_Value
[82] SI_Target_Rating_Chamber_Leakage_Estimation_Topleft_Value_Smoothed
[83] SI_Target_Rating_Chamber_Leakage_Estimation_Bottomleft_Value
[84] SI_Target_Rating_Chamber_Leakage_Estimation_Bottomleft_Value_Smoothed
[85] SI_Target_Rating_Chamber_Leakage_Estimation_Bottom_Value
[86] SI_Target_Rating_Chamber_Leakage_Estimation_Bottom_Value_Smoothed
[87] SI_Target_Rating_Chamber_Leakage_Estimation_Bottomright_Value
[88] SI_Target_Rating_Chamber_Leakage_Estimation_Bottomright_Value_Smoothed
[89] SI_Target_Rating_Chamber_Leakage_Estimation_Center_Value
[90] SI_Target_Rating_Chamber_Leakage_Estimation_Center_Value_Smoothed
[91] SI_Rating_Of_Adhesion_Force [92] SI_Rating_Of_Adhesion_Tilt
[93] SI_Rating_Of_Adhesion
A.8.1 Weights Used in Simulation Environment

Based on the described optimization process, the following sets of weights have been acquired in the simulation environment. The first one given in listing 3 contains the 90 weights used to evaluate the 90 behavior values while driving down the wall.

Listing 3: Trained weights for downward motion.

```
0.225775, -0.96465, -0.358468, -0.505392, 0.300171, 0.170662, 0.649598,
0.91231, 0.429061, -0.391973, 0.452485, -0.413515, 0.0268819, -0.391204,
0.179616, -0.599079, 0.864802, -0.461733, -0.915699, 0.874346, -0.302394,
-0.670615, -0.769278, 0.93863, -0.519058, 0.22361, 0.188837, 0.742087,
0.496084, -0.887774, 0.764858, -0.503325, 0.376558, 0.0839427, 0.012354,
0.203776, 0.0792224, 0.891382, 0.430042, -0.584446, 0.407561, 0.0150046,
-0.607849, 0.179846, 0.683777, -0.643492, 0.204954, -0.290895, -0.290927,
0.125018, 0.168553, -0.191239, -0.336198, -0.587588, -0.0200349, -0.358533,
0.759818, -0.456328, 0.750117, -0.830985, 0.707035, -0.353054, 0.275749,
-0.157087, 0.999165, 0.99727, 0.773886, 0.0703945, 0.664542, -0.619196,
0.592087, 0.241767, 0.296139, 0.637239, -0.920751, -0.138244, 0.597345,
-0.435909, -0.300285, 0.227492, 0.03705, 0.159642, 0.168787, 0.821414,
0.338637, -0.921364, -0.312215, 0.985323, -0.188447, 0.0694116
```

Listing 4 contains the weights used for upward motion. They have been acquired in the same way as in the previous case by the described optimization process based on several training sets. As described in section 8.4.5, it is not possible to get weights fitting for these oppositional situations, which makes it necessary to determine different sets of weights for varying cases.

Listing 4: Trained weights for upward motion.

```
0.836746, 0.394634, -0.607269, -0.528252, -0.388559, 0.472662, 0.195135,
0.924691, 0.0295141, -0.309328, -0.0497406, 0.787647, 0.298908, 0.203738,
-0.0941318, 0.726185, -0.460141, -0.691011, 0.617703, -0.554489, 0.029301,
-0.491998, -0.322662, 0.183839, -0.86125, -0.107175, -0.185828, -0.903024,
-0.0442661, 0.848716, 0.376531, 0.229557, -0.160242, -0.417986, -0.647737,
0.933477, -0.266583, -0.551228, 0.0487016, -0.173237, -0.811268, 0.376234,
-0.074378, 0.744176, 0.995542, -0.0170062, -0.278161, 0.118372, 0.376399,
0.0835435, 0.664409, -0.251017, -0.642244, 0.663157, 0.709232, -0.52442,
-0.8827, 0.489555, -0.11735, 0.171179, -0.87677, -0.325339, 0.39984,
0.999965, 0.19648, -0.611864, -0.866399, 0.503412, 0.13198, 0.532757,
0.949162, 0.99961, -0.342302, -0.467621, -0.999926, -0.999435, -0.293359,
-0.389773, -0.158754, 0.456151, -0.133449, -0.373818, 0.161545, -0.654764,
-0.665934, -0.00320321, -0.342873, 0.228968, 0.568053, 0.520553
```

The last type of robot motion – sideward driving – is considered by the weights given in listing 5. Because of the geometric setup of the robot there is no distinction needed for driving left or right, so a common set of weights could be acquired.
Listing 5: Trained weights for sideward motion.

\[-0.747702, 0.921862, 0.614635, 0.399011, -0.390782, 0.786466, 0.0139231,\]
\[-0.062903, -0.177223, 0.572824, -0.589295, 0.169648, 0.378936, -0.77695,\]
\[0.361959, 0.186227, 0.472622, 0.3197, 0.431333, -0.796842, -0.308438,\]
\[-0.939519, -0.0805065, 0.834158, -0.526049, -0.272676, -0.183844, -0.504856,\]
\[-0.621139, 0.275426, 0.625826, 0.221891, 0.197622, 0.061364, -0.312719,\]
\[0.165347, -0.132228, 0.327297, 0.095396, -0.416005, -0.586309, -0.434335,\]
\[-0.378623, -0.665646, 0.246843, 0.444024, 0.0911544, -0.029188, -0.222015,\]
\[0.445364, 0.195107, 0.782834, 0.140599, 0.356721, 0.190895, 0.722385,\]
\[-0.177009, -0.0475243, 0.693467, -0.622466, -0.418963, 0.226115, -0.52666,\]
\[0.419839, -0.697527, -0.492283, -0.699806, -0.106867, 0.849364, -0.704367,\]
\[-0.42852, 0.185212, -0.0855719, -0.37405, -0.370934, 0.0875872, -0.244485,\]
\[-0.247706, -0.544572, -0.824705, 0.405017, -0.201544, 0.264545, -0.567075,\]
\[0.0184378, 0.110131, 0.305538, 0.040284, -0.291689, -0.153153,\]

A.8.2 Weights Applied on Real Prototype

Listing 6 shows the set of weights, which has been used during the real experiments presented in section 9.3. Due to limitations of the robot hardware, only one general set of weights based on a handful of training sets has been acquired and used for validating the presented approaches.

Listing 6: Trained weights for the real world experiments.

\[0.0866154, -0.484857, -0.510924, 0.300484, -0.687718, -0.263489, 0.54117,\]
\[-0.327868, -0.615799, -0.0760936, 0.191637, 0.00782626, -0.68406, -0.620042,\]
\[-0.919995, -0.160523, 0.906961, 0.814032, -0.266099, 0.48363, 0.906739,\]
\[0.477688, 0.830664, 0.0970128, 0.384372, -0.375302, 0.605658, 0.518466,\]
\[-0.591144, 0.940376, 0.0823173, 0.22747, -0.90883, 0.50833, 0.18048,\]
\[-0.571361, 0.81402, -0.212767, 0.0724666, 0.186345, 0.133371, -0.311728,\]
\[-0.190242, -0.994126, 0.0994504, -0.419223, -0.171059, 0.634438, 0.989271,\]
\[0.332636, 0.101427, -0.135896, 0.363566, -0.0205128, 0.578462, -0.268265,\]
\[0.271416, -0.944133, 0.109213, -0.493751, 0.914361, 0.765587, 0.681198,\]
\[0.335499, 0.937305, 0.745464, -0.999175, -0.998425, 0.991362, 0.225146,\]
\[-0.999137, -0.997563, 0.873475, -0.423103, -0.999814, -0.999734, 0.198967,\]
\[-0.586114, -0.103005, 0.237946, -0.607879, -0.752111, 0.410675, 0.909159,\]
\[-0.429169, 0.297608, 0.0104689, -0.167144, -0.474464, 0.684448,\]
A.9 Optimization Process

This section presents details of the optimization process experiments using different parameters and settings for training (compare section 8.3). The results are presented as rating values $R$ which have to reach at least the break point $R_{bp}$ based on the index number of training sets. Above this point (dashed line in the following figures) the rating function delivers a satisfying result. The first part contains different characteristics based on changes of mutation parameters and elite settings. Afterwards, the influence of large populations on the optimization process will be shown in detail. The last part presents plots with too different training sets.

A.9.1 Mutation Parameters

Figure A.18 shows experiments, which are similar to figure 8.13 with one difference: The chance of elite injection is reduced to $p_e = 0.001$ resulting in slower and less smooth optimization process.

![Figure A.18](image)

Figure A.18: Optimization process with smaller elite chance using $|P| = 100$, $p_{off} = 0.75$, $p_{mand} = 0.5$, $p_{mud} = 0.1$, $p_e = 0.001$ and ten training sets.

The plots in figure A.19 show the optimization process using only offset mutation as genetic operation. Compared to the standard parameters given in figure 8.13 the general optimization process is a bit slower. Figure A.20 shows the optimization process using only random weight mutation as genetic operation. It can be seen that the results are similar to the previous mutation settings (figure A.19) but with a smaller bandwidth of different evaluation ratings. Normed mutation parameters do not lead to significant different results, as given in figure A.21. The same counts for figure A.22 which has a higher chance of offset mutation.
Figure A.19: Optimization process with offset mutation only ($|P| = 100, p_{m}^{\text{off}} = 1.0, p_{m}^{\text{rand}} = 0.0, p_{m}^{\text{mul}} = 0.0, p_{e} = 0.1$, ten training sets).

Figure A.20: Optimization process with random mutation only ($|P| = 100, p_{m}^{\text{off}} = 0.0, p_{m}^{\text{rand}} = 1.0, p_{m}^{\text{mul}} = 0.0, p_{e} = 0.1$, ten training sets).
Figure A.21: Optimization process using normed mutation parameters $|P| = 100, p_{m}^{off} = 0.5, p_{m}^{rand} = 0.5, p_{m}^{mul} = 0.0, p_e = 0.1$ and ten training sets.

Figure A.22: Optimization process with higher mutation rate using $|P| = 100, p_{m}^{off} = 1.5, p_{m}^{rand} = 0.5, p_{m}^{mul} = 0.0, p_e = 0.1$ and ten training sets.
A.9.2  Population Size

The influence of the population size on the speed of the optimization process can be seen in figure A.23. Here the optimization runs were executed using 1000 individuals per population. The optimization process is slower compared to the experiments with only 100 individuals.

![Figure A.23](image)

**Figure A.23:** Optimization process using larger populations ($|P| = 1000$, $p_{off}^{off} = 0.75$, $p_{m}^{rand} = 0.5$, $p_{m}^{mul} = 0.1$, $p_{e} = 0.01$) and ten training sets.

The results of using 10000 individuals per population are depicted in figures A.24 and A.25 using low and normal rate of elitism. Although the rating values start better compared to experiments with less individuals the optimization process is very slow.

![Figure A.24](image)

**Figure A.24:** Optimization process with large population of 10000 individuals and low elitism rate ($|P| = 10000$, $p_{off}^{off} = 0.75$, $p_{m}^{rand} = 0.5$, $p_{m}^{mul} = 1.0$, $p_{e} = 0.001$, ten training sets).
A.9.3 Contradictive Training Sets

Test runs trying to find suitable weights for driving up and driving down are shown in figure A.26. The break point is not reach by any of the experiments, no weights could be found which fulfill the desired functionality.

Figure A.26: Optimization process trying to find weights for driving upwards and downward (\(|P| = 100, p_{off}^{p} = 0.5, p_{rand}^{p} = 0.5, p_{mul}^{p} = 0.0, p_{e} = 0.1, 15 training sets\).
Bibliography


tional Safety”, Networks and Distributed Systems, Bielefeld University, Bielefeld, Ger-


K. Berns, “Design and development of a sealing-system for climbing-robots”, in Pro-
ceedings of the Gepeszet Mechanical Engineering Conference. Budapest, Hungary, 2008,
pp. 1–10.

WallClimbing Robot”, Chinese Journal of Mechanical Engineering, vol. 23, no. 2007,

Driven Climbing Robot”, in Proceedings of the International Conference on Robotics


of the Art and Experimental Considerations”, in Proceedings of the 11th International
Conference on Climbing and Walking Robots (CLAWAR). Singapore: World Scientific

Climbing Service Robot For Remote Inspection And Maintenance In Hazardous Envi-
ronments”, in Proceedings of the 8th IEEE Conference on Mechatronics and Machine

climbing service robot for remote maintenance applications in hazardous environments”,

robots for safety inspection of nuclear reactor pressure vessels”, Journal of Measurement

climbing and mobile service robots for remote inspection and maintenance in nuclear


Index

$(\mu + \lambda)$-strategy, 177
accuracy, 195, 238
activation, 28
active suction chambers, 34
activity, 28
adhesion, 18, 32
adhesion chamber, 19
adhesion control, 21, 113
adhesion control network, 153
adhesion system, 18
advanced motion control, 95
air flow control, 119
ALARP, 45
ambiguities, 97
AMC, 95
antilock brake system, 100
applied downforces, 143
artificial neural networks, 172
attack tree, 50
backward kinematic, 14
basic adhesion control, 21, 116
basic fault event, 78
behavior, 27
activation, 28
activity, 28
inhibition, 28
stimulation, 28
target rating, 28
behavior fusion, 28
behavior-based architecture, 27
behavior-based control, 27, 113, 208, 215
behavior-based framework, 27
behavioral group, 29
blackboard, 257
boolean expressions, 240

CAED, 51, 252
case-dependent generic function, 241
cause and effect diagram, 51, 252
chamber control, 118
chamber deactivation, 117, 123, 127
chamber pressure calculation, 142
chamber pressure control, 119
chamber reintegration, 128, 132
children, 178, 184
chromosome, 174, 178, 184
clamping, 35
claws, 35
climbing robots, 9, 31
collision, 60
collision avoidance, 61
composed event, 78
counteractive measures, 95, 201, 206, 211
CROMSCI, 9, 217
crossover, 175, 184
data collection, 74
dead reckoning, 17
defect, 79
DETAM, 50, 51, 254
digital signal processor, 12, 95
direct kinematic, 15
draw back, 213
drive controllers, 12, 101
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
51, 254
drive system, 13
driving resistance, 66
driving strategy, 216
DSP, 12, 13, 95
dynamic errors, 82
dynamic event tree analysis method, 50,
elitism, 177, 186
emergency stop, 74, 206, 211
environmental sensors, 25
evasion, 206
event tree, 254
evolutionary algorithm, 174
expert knowledge, 74
external hazard, 60

failure, 79
failure date, 74
failure mode and effect analysis, 51, 52, 252
failure rates, 82
fallout, 238
false negative, 237
false negatives, 187, 195
false positive, 187, 195, 237
fault, 79
fault tree, 50, 52, 78
fault tree analysis, 51, 78
fishbone diagram, 252
fitness function, 174
FMEA, 50–52, 252
force control, 117, 139
force point control, 141
force value control, 140
forward kinematic, 15
friction, 65
friction circle, 65
friction coefficient, 65
FTA, 50, 51
fusion behavior, 28
gecko principle, 36
generic function, 240
genetic algorithm, 174, 178
genitor algorithm, 177
geometry blackboard, 262
grasping, 36
gridmap, 262
gripping, 35
group behavior, 29
GUI, 258

hardware measures, 77, 87
hardware requirement, 87
hazard, 41, 58
hazard and operability studies, 49
hazard classification scheme, 42
hazard identification techniques, 49
high consequence robotics system, 52
hole, 62
hybrid learning, 172
IB2C, 27
inactive chamber, 123
individual, 174, 178
industrial robot, 45
inhibition, 28
inspection sensors, 24
inspection system, 24
interaction model, 74
interface safety analysis, 51, 253
internal hazard, 59
internal sensors, 25
inverse kinematic, 14
ISA, 51, 253
kinematic constraints, 244
kinematics, 14
laser range sensor, 25, 202
laser scanner, 25
leakage estimation, 124
learning, 172
legged motion, 32
localization, 202
locomotion, 11
macro structures, 60, 201
magnetic adhesion, 32
magnets, 32
malfunction, 78
manipulator, 24
map, 262
markov analysis method, 253
markov method, 51
massflow, 119
maximum function, 239
mca, 256
mca browser, 257
mca gui, 258
mean time to failure, 81, 254
mean-square average, 183
mechanical adhesion, 35
meta data, 27, 113  
micro structures, 70  
miss rate, 238  
module, 256  
motion control, 11, 95, 208  
motion control network, 215  
motor controller, 12, 101  
MTBF, 81, 82  
MTTF, 81, 82, 254  
Murphy’s Law, 1  
mutation, 175  

near-to-far learning, 211  
negative pressure adhesion, 19, 33  
negative pressure system, 19  
near-to-far learning, 211  
negative pressure adhesion, 19, 33  
negative pressure system, 19  
nuclear safety cross-check analysis, 51, 253  

obstacle avoidance, 74, 206  
obstacle detection, 203  
obvious risk, 81  
odomtery, 17, 202  
offline analysis, 75, 78  
omnidirectional drive, 13  
online analysis, 78  
online requirements, 89  

passive suction cups, 33  
perception, 74  
PFD, 254  
physic simulation, 266  
PhysX engine, 266  
PID, 12  
pitch, 63  
POI, 203  
point of downforce, 69, 141  
point of interest, 203  
policy of guaranteed safety, 47  
population, 174, 178  
position control, 12  
power failure, 80, 81  
predictable hazards, 201  
pressure control, 119  
pressure trend, 125  
primary event, 78  
probability of fail on demand, 254  
probability tree analysis, 51, 253  
propulsion, 11  
pure random search, 192  

qualitative description, 49  
quantitative description, 49  
random search, 192  
random walk, 173  
redus controller, 13  
reinforcement learning, 74, 173  
reintegration of chambers, 128, 132  
requirements, 87, 91  
reservoir chamber, 19  
reservoir control, 121  
risk, 42  
risk assessment, 46, 50, 252  
risk estimation, 73, 75  
risk graph, 252, 255  
risk graph method, 51, 52, 254  
risk handling, 50, 201  
risk parameters, 52  
risk prediction, 73  
robot slip, 70, 80, 85  
robot tilt, 68, 80, 85  
roll, 63, 245  
roll-pitch-yaw, 245  
rolling constraints, 15, 244  
rotation matrix, 238  
roulette wheel method, 174  
safe navigation, 73, 95, 163  
safe state, 74, 213  
safeguarding, 55, 251  
safety, 43  
safety analysis, 49, 252  
safety concept, 92  
safety critical system, 44  
safety fault tree, 50  
safety guidelines, 44  
safety integrity level, 53, 251, 254  
safety levels, 44  
safety measures, 46, 47, 60, 73–75, 252  
safety requirement, 45, 87  
sealing, 23  
sealing abrasion, 68  
sealing defect, 85
Index

sealing leakage, 68
sealing simulation, 266
secondary event, 79
sector map, 203
security attack tree, 50
self-safety, 47
sensitivity, 238
sensors, 25
severity classes, 42, 58
SFC, 67, 105
shear force controller, 67, 105
shear forces, 67, 105
sideward force, 105
sil, 53, 251, 254
simulated annealing, 173
simulation, 262
simvis 3d, 262
situation awareness, 53
situation operator model, 55
sliding constraints, 14, 244
sliding frame, 32
slow down, 206
SOM, 55
specificity, 238
spines, 35
static obstacles, 201
steerable standard wheel, 11
sticky tapes, 36
stimulation, 28
suction cups, 33
supervised learning, 172
surface analysis, 25, 232
target rating, 28
tCS, 67, 100
technic of operations review, 50, 51, 253
terrain characteristics, 176, 232
terrain classification, 2, 232
test wall, 218
tetris, 177
thermal glue, 36
thermodynamic model, 267
thermodynamical model, 74
tilt, 63, 245
TOR, 50, 51, 253
traction, 100
traction control system, 67, 89, 100
training data, 171, 178
trigger event, 79
true negative, 237
true negatives, 195
true positive, 195, 237
ubiquitous risk, 81
unsupervised learning, 172
Van der Wals forces, 36
velocity control, 13
vortex, 33
VRAM, 33
watchdog, 53, 55
weighted maximum function, 239
wheel, 11
wheel abrasion, 65
wheel overturning, 97
wheel slip, 65, 67, 97, 100
working chamber, 19