ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Bachelor-Thesis

Obstacle Climbing Control

for an Inspection Robot
with Magnetic Wheels

Spring Term 2013

Supervised by:
Philipp Krüsi
Dr. Martin Rufli
Prof. Roland Siegwart

Author:
Florian Berlinger

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Obstacle Climbing Control for an Inspection Robot with Magnetic Wheels

is original work which I alone have authored and which is written in my own words.[1]

Author

Florian	Berlinger

Supervising lecturer

Philipp	Krüsi

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`http://www.ethz.ch/students/exams/plagiarism_s_en.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date	Signature

[1]Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Foreword

This bachelor thesis was written at ETH Zurich during the spring semester 2013. The impulse to do so was my participation in the focus project Ship Inspection Robot, which needed a motion control for its obstacle climbing robot.

First, I would like to express my gratitude to my supervisors, Philipp Krüsi and Dr. Martin Rufli, not only for their support and advice during my research, but also for their acceptance and encouragement of my self-initialised thesis proposal.
Thanks as well to Prof. Roland Siegwart, head of the Autonomous Systems Lab at ETH Zurich, for giving me the opportunity to write my bachelor thesis there.

Additionally, I would like to thank Alstom Inspection Robotics, the industrial partner of Ship Inspection Robot, for their financial support for the realisation of an authentic testing environment.

Moreover, I very much appreciate the love, enthusiasm and moral encouragement, which friends and family have shown to me.

Finally, thank you team SIR. To design the Ship Inspection Robot together with you was the best experience during my bachelor's degree at ETH Zurich. I am deeply grateful for having enjoyed such an inspiring and emotional time together with you.

Contents

Abstract

Obstacle Climbing Control pursues two main objectives in order to enable robots to overcome obstacles in a sound way. On the one hand, it reduces the wheel slip. Therefore, Obstacle Climbing Control allows the precise drive of trajectories and prevents potential damage of the surface. On the other hand, if there is no wheel slip, it minimises the structural stress on the drivetrain and thus enhances the lifetime of the robots.
The need for Obstacle Climbing Control arises from the geometrical fact that the front and the rear wheels of robots need to turn at different speeds whenever the environment is not flat, e.g., when climbing an obstacle.

The first implementations and experiments with Obstacle Climbing Control are carried out on the four magnetic wheel driven Ship Inspection Robot. Ship Inspection Robot inspects the inner hull of big bulk carriers for corrosion and rust, where it faces several types of obstacles.
Experiments show that wheel slip is almost negligible for Ship Inspection Robot. Consequently, the test runs prove that obstacles cause high torque peaks. Obstacle Climbing Control can reduce these torque peaks by up to 40 %.

The torque reduction is achieved in two ways: First, with a feedforward controller which is based on geometrical calculations; second, with a feedback controller which compares the front and the rear wheel torques in order to adjust the wheel speeds correctly.
The newly designed feedback controller has two advantages over the feedforward approach and over the state of the art, namely additionally mounted resistive wire strains or optical encoders. Neither does it require the identification of obstacles nor is it necessary to mount additional measurement devices for the detection of wheel slip or structural stress.

Key words: Obstacle climbing, ship inspection, magnetic wheel, inspection robot, SIR, focus project

Symbols

Symbols

$a = 50$	absolute shaft centre distance $[mm]$
$r = 30$	wheel radius $[mm]$
$l = 110$	robot length $[mm]$
$l_f = 25$	front wheel length distance to the COM $[mm]$
$l_b = 25$	rear wheel length distance to the COM $[mm]$
$w_f = 160$	front wheel width distance to the COM $[mm]$
$w_b = 110$	rear wheel width distance to the COM $[mm]$
f	front wheel distance to the ICR $[mm]$
b	rear wheel distance to the ICR $[mm]$
x	horizontal shaft centre distance $[mm]$
y	vertical shaft centre distance $[mm]$
s	vertical circle distance $[mm]$
l	chord length $[mm]$
a_{pre}	covered front wheel distance $[mm]$
a_{sym}	covered front wheel distance at the symmetry point $[mm]$
γ	obstacle angle $[rad]$
α	running angle for wheels on convex edges $[rad]$
v_x	robot absolute speed $[mm/s]$
v_f	front wheel speed $[mm/s]$
v_b	rear wheel speed $[mm/s]$
n_f	front wheel revolutions [-]
n_b	rear wheel revolutions [-]
t_i	phase times inside obstacles $[s]$

Indices

f	front
b	back/rear

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
ASL	Autonomous Systems Lab
SIR	Ship Inspection Robot
OCC	Obstacle Climbing Control
CHF	Swiss franc
IMU	Inertial Measurement Unit
4WD	four-wheel drive
DC	direct current
rpm	revolutions per minute
ICR	instantaneous centre of rotation
COM	centre of mass
MC	motor controller

Chapter 1

Introduction

The key challenge for the Ship Inspection Robot[1] is to overcome obstacles like bulb-, L-, I- or T-profiles, as well as convex and concave edges. By doing so, SIR requires different drives for front and rear wheels due to the geometrical constraints of the uneven environment. More precisely, the front and the rear wheels cover different distances within the same time (cf. Figure 1.1).
Neglecting these drive ratios would result in either wheel slip and potential damage of the surface to be inspected or in undesirable moments and forces acting on the robot's structure in the case of no wheel slip which is expected for SIR.

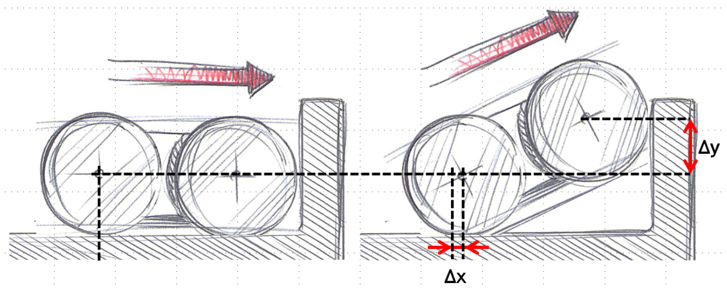


Figure 1.1: Reason for Obstacle Climbing Control.

SIR is driven by four DC-motors - one on each wheel. The motors display their currently used voltages and currents/torques as well as their drives and positions. Infrared and gyro sensors, an IMU and the mounted GoPro camera provide further control-relevant information. The control software is installed on a base station, which is wired with the robot in order to send the control commands quickly.
The geometry of (see Figure 1.2) and the acting forces on the skid-steered Ship Inspection Robot as well as an overview on the software structure are discussed in Chapter 4.2.
This thesis includes the development and implementation of a suitable control strategy to overcome obstacles, testing on the real system, as well as a detailed evaluation of the results.

1.1 State of the art in slip and torque reduction

SIR is the first inspection robot of its type, which has the ability to climb a broad variety of obstacles thanks to its overlapping magnetic wheel configuration and its

[1]Ship Inspection Robot (SIR) was designed within a focus project at ETH Zurich during the academic year 2012/2013. [1]

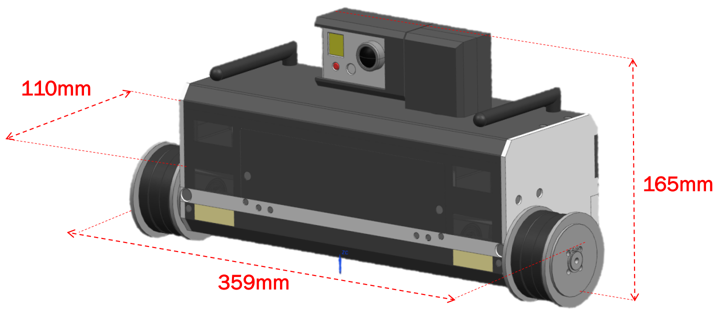


Figure 1.2: Dimensions of SIR.

lever mechanism. Current solutions are mostly unable to avoid the deformation of a vehicle inside obstacles or the scratching of the surface because of wheel slip. One exception is the two-wheeled MagneBike[2]. It uses a resistive wire strain to minimise the deformation of its structure.
However, anti-slip controllers of modern cars like the Anti-lock Brake System (ABS) or the Electronic Stability Control (ESC) fulfil similar tasks. Therefore, their adaptability for a usage on SIR is checked in Chapter 4.1.3.

1.2 Contributions towards electrically driven vehicles

This thesis makes three major contributions. Firstly, it enables SIR to climb obstacles in a safe way. This objective is achieved by either a reduction in wheel slip or structural stress. The implemented controller ideally also prevents SIR from falling off uneven walls.
Secondly, this thesis discusses a new approach for an anti-slip controller for four-wheel electrically driven vehicles with respect to the state of the art.
Finally, a testing ground similar to the real environment inside the ships, which have to be inspected by SIR, is designed.

1.3 Structure of this thesis

According to the contributions, this thesis is split into three main parts.
Chapter 2 briefly describes the design of a suitable testing environment in order to test the implemented controllers on the robot.
Chapter 3 addresses a feedforward control solution for edges of different angles that have to be known.
Chapter 4 focuses on the design of a feedback controller which does not need any information about the type of the obstacle.
Finally, the conclusion summarises the thesis. It shows the limits of the chosen control strategies as well as recommendations for future developments.
For interested readers, the Appendix offers additional and more detailed background insight.

[2]The MagneBike was developed at ETH Zurich as well. It is designed for power plant inspection and optimised for the movement inside tubes. [2]

Chapter 2

Testing environment

Ship Inspection Robot is designed for the inspection of the world's biggest bulk carriers, which obviously do not cruise in Switzerland. Therefore, the design of a testing environment, as similar to the real cargo holds as possible, is important to gain quick testing results.
Additionally, Chapter 2 gives an overview of the obstacles which SIR has to overcome.

2.1 Real operating environment

According to ship plans and pictures of cargo holds, the most common obstacle profiles are defined and dimensioned (Figure 2.1 (right), f.l.t.r: I-, bulb-, L- and T-profile).

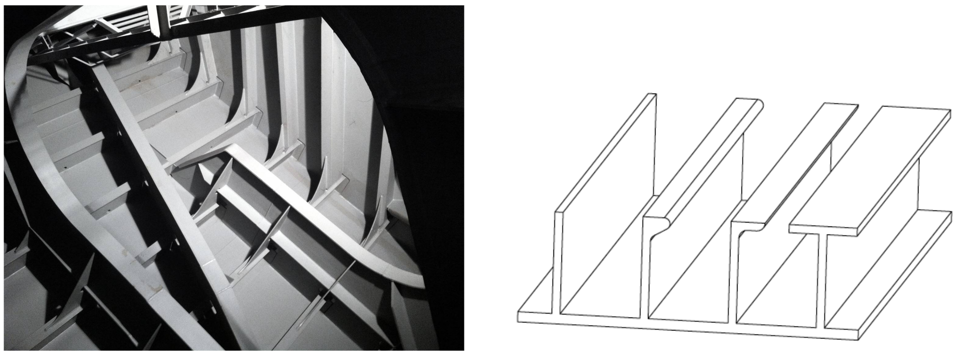


Figure 2.1: The inside of a bulk carrier and its deduced profiles.

The coated steel-plates are usually covered with corrosion and oil waste. For the experiments with the robot prototype, laboratory conditions are defined. The steel plates which are used for the testing environment are rust-free, clean and uncoated. These properties result in better wheel-adherence and less damage of the robot due to dirt.

2.2 Design of a testing environment

A testing environment for experiments on Ship Inspection Robot has to take the following aspects into consideration:

- Costs

- Reusability for further inspection robots
- Safety
- Mobility
- Size
- Mass

After several negotiations, SIR's industrial partner grants an additional budget of CHF 1750.- for testing. To be able to test and present the running robot in various locations, a wheeled framework, made out of item-profiles, is designed. The actual testing ground includes all the defined profile types in real scale and is installed on top of the framework. It is rotatable so that different influences of gravity can be tested. Four holes in the main steel-plate allow to fix additional obstacles (Figure 2.2 (right)). The whole testing environment has a mass of around 100 kg. With a size of 800x900x1000 [mm], it is big enough to let the robot turn-on-spot on the main plate, but also small enough to fit through standard doors (Figure 2.2 (left)). In order to detach the robot safely from the magnetic steel plates, a special ramp made out of wood and aluminium is designed as well. Stability calculations and technical drawings are shown in Appendix A.

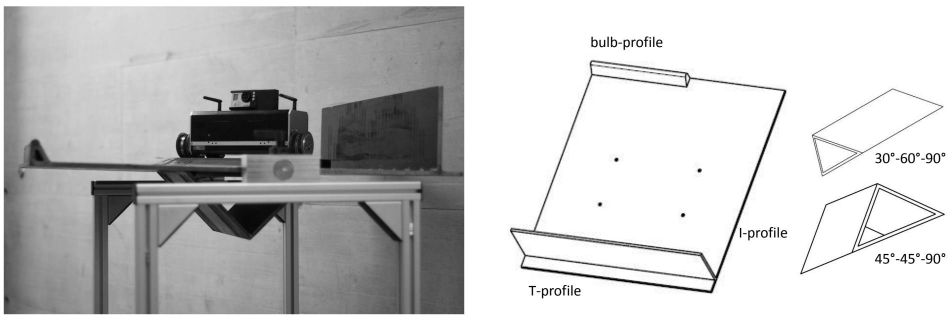


Figure 2.2: Characteristics of the testing environment.

Chapter 3

Feedforward control approach based on geometrical considerations

The feedforward approach is based on geometrical calculations depending on the obstacle. For a constant robot absolute speed v_x on the leading wheels, the ideal speed ratio for the controlled following wheels is defined by using the instantaneous centre of rotation. The two front wheels turn with equal speeds as well as the two rear wheels. This is important to ensure a straight motion over obstacles in order to prevent the robot from falling off.
Different angles of the connected plates from $\gamma = \frac{\pi}{2}$ to $\gamma = \frac{3}{2}\pi$ need to be taken into account. The angles γ are considered to be given inputs, measured by SIR's infrared sensors. Fortunately, the standard angles for the obstacles inside ships are either $\gamma = \frac{\pi}{2}$ (concave edge) or $\gamma = \frac{3}{2}\pi$ (convex edge).
The starting times t_0 for the speed ratio algorithms have to be defined and detected for all cases. This is relatively easy in the concave case. SIR's lever mechanism detaches the front wheels from the steel plates. Therefore, their torques drop and their drives rise. The convex case is more complex as the lever does not take action there. Experiments should expose how to detect the starting time in the convex case. Using an IMU to detect the change in angle would be an additional option. However, this would result in a significant delay to start the feedforward algorithm.

3.1 Derivation of speed profiles

This chapter addresses the feedforward speed ratios to different obstacles and provides an overview of the results. The detailed calculations for each case are presented in Appendix B.

3.1.1 Concave edges

Concave edges are defined as $\gamma \in [\frac{\pi}{2}, \pi]$. Inside concave edges, the front wheels take the lead and run at the constant robot absolute speed, so $v_f = v_x$. The rear wheel speeds v_b for a general obstacle angle γ are (cf. equation B.2)

$$v_b = v_f \cdot \left(-cos\gamma + \frac{v_f \cdot \ t \cdot sin^2\gamma}{\sqrt{a^2 - (v_f \cdot \ t \cdot sin\gamma)^2}} \right) \tag{3.1}$$

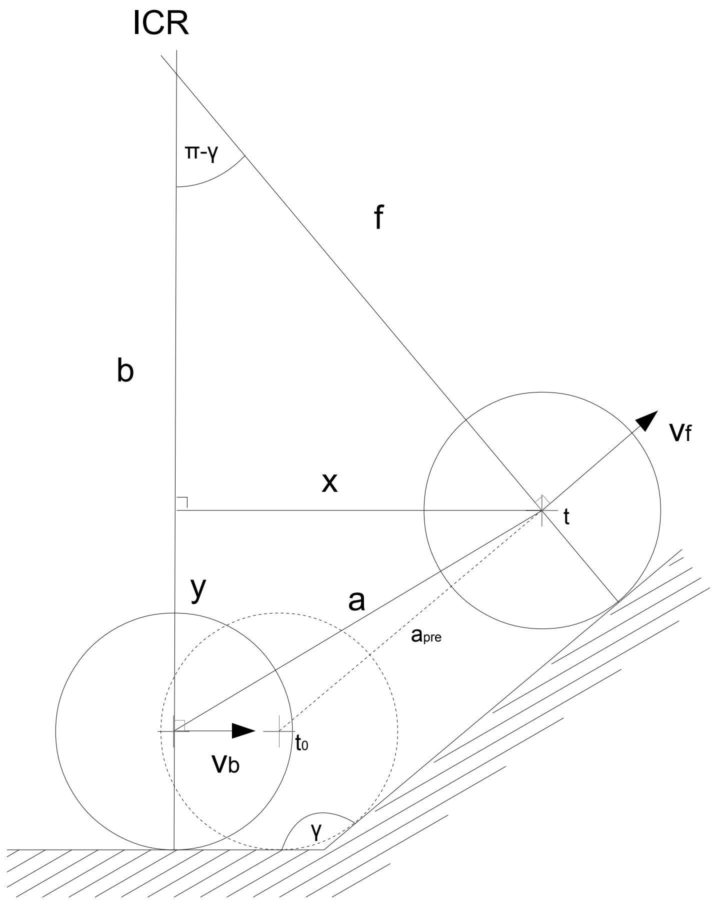


Figure 3.1: Concave edge climbing situation.

In the widespread case of $\gamma = \frac{\pi}{2}$, the rear wheel speeds v_b are

$$v_b = \frac{v_f^2 \cdot t}{\sqrt{a^2 - (v_f \cdot t)^2}} \tag{3.2}$$

The speed ratios are shown in Figure 3.3 (left).

3.1.2 Convex edges

Convex edges are defined as $\gamma \in [\pi, \frac{3}{2}\pi]$. Their traverse is split in four phases (Figure 3.2 (left) shows them and the points 1 and 2) which in the leading wheels are always driving at the constant absolute robot speed v_x:

- Phase 1 from t_0 to t_1: The front wheels are at constant speeds between point 1 and 2 (the wheel centre is relevant); the rear wheel speeds have to be adjusted.
- Phase 2 from t_1 to t_2: All wheels are outside the sector between point 1 and 2 (general situation before "halftime"); the rear wheel speeds have to be adjusted.

- Phase 3 from t_2 to t_3: All wheels are outside the sector between point 1 and 2 (general situation after "halftime"); the front wheel speeds have to be adjusted.

- Phase 4 from t_3 to t_4: The rear wheels are at constant speeds between point 1 and 2; the front wheel speeds have to be adjusted.

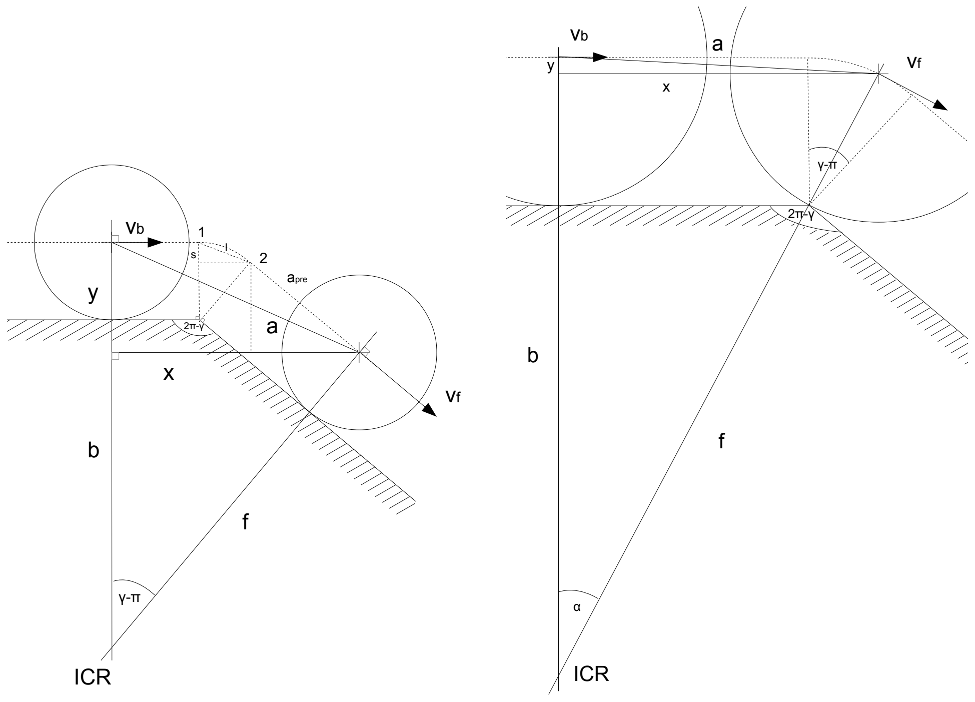


Figure 3.2: Convex edge climbing situations.

Thanks to the symmetry of the edges, the calculations have to be done only for the first half of the obstacle (phases 1 and 2). The front wheels start as leaders with the absolute robot speed v_x. The rear wheel speeds are adjusted accordingly until "halftime" t_2. At this point, the lead is given to the rear wheels, which now turn at v_x, and the front wheel speeds are adjusted. The calculations of the phase times t_0 to t_4 are shown in equation B.6.

For the general situation before "halftime" (phase 2), the speed calculations are similar to the concave case and lead to (cf. equation B.3)

$$v_b = v_f \cdot \left(-cos\gamma - sin\gamma \cdot \frac{-v_f \cdot \ (t - t_1) \cdot sin\gamma + 2r \cdot cos^2(\frac{\gamma}{2})}{\sqrt{a^2 - (-v_f \cdot \ (t - t1) \cdot sin\gamma + 2r \cdot cos^2(\frac{\gamma}{2}))^2}} \right) \tag{3.3}$$

and for $\gamma \ = \ \frac{3}{2}\pi$:

$$v_b = v_f \cdot \frac{v_f \cdot \ (t - t_1) + r}{\sqrt{a^2 - (v_f \cdot \ (t - t_1) + r)^2}} \tag{3.4}$$

In phase 3 after "halftime" but still during the general situation, the speeds are the following (cf. equation B.7):

$$v_f = v_b \cdot \left(-cos\gamma - sin\gamma \cdot \frac{-v_b \cdot \ (t_3 - t) \cdot sin\gamma + 2r \cdot cos^2(\frac{\gamma}{2})}{\sqrt{a^2 - (-v_b \cdot \ (t_3 - t) \cdot sin\gamma + 2r \cdot cos^2(\frac{\gamma}{2}))^2}} \right) \tag{3.5}$$

or for $\gamma \ = \ \frac{3}{2}\pi$:

$$v_f = v_b \cdot \frac{v_b \cdot \ (t_3 - t) + r}{\sqrt{a^2 - (v_f \cdot \ (t_3 - t) + r)^2}} \tag{3.6}$$

If one wheel pair is between point 1 and 2 (Figure 3.2 (right)), a new situation arises.
In phase 1, the rear wheel speeds v_b are (cf. equation B.5)

$$v_b = v_f \cdot \left(cos(\frac{v_f \cdot \ t}{r}) + sin(\frac{v_f \cdot \ t}{r}) \cdot \frac{2r \cdot sin^2(\frac{v_f \cdot \ t}{2r})}{\sqrt{a^2 - \left(2r \cdot sin^2(\frac{v_f \cdot \ t}{2r})\right)^2}} \right) \tag{3.7}$$

In phase 4, the front wheel speeds v_f are (cf. equation B.8)

$$v_f = v_b \cdot \left(cos(\frac{v_b \cdot \ (t_4 - t)}{r}) + sin(\frac{v_b \cdot \ (t_4 - t)}{r}) \cdot \frac{2r \cdot sin^2(\frac{v_b \cdot \ (t_4 - t)}{2r})}{\sqrt{a^2 - \left(2r \cdot sin^2(\frac{v_b \cdot \ (t_4 - t)}{2r})\right)^2}} \right) \tag{3.8}$$

The speed ratios for all four phases are shown in Figure 3.3 (right).

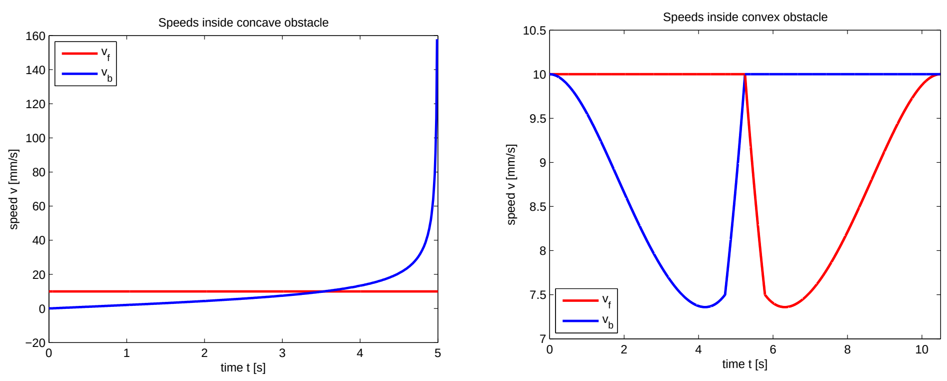
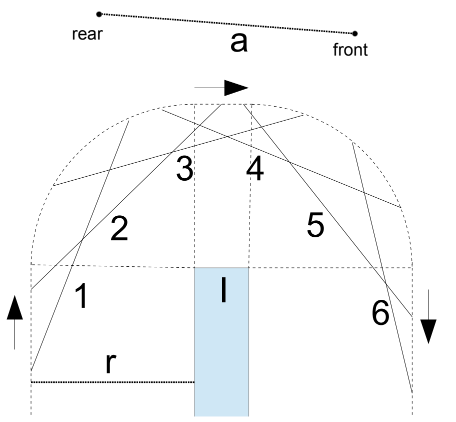


Figure 3.3: Speed ratios for $\gamma = \frac{\pi}{2}$ (concave) and $\gamma = \frac{3}{2}\pi$ (convex).

3.1.3 I-profiles

The traverse of I-profiles is split in six phases (Figure 3.4 (left)). The front wheels lead within the first three phases. Then, symmetry is used again and the rear wheels lead within the symmetrical phases 4, 5 and 6. Fortunately, I-profiles are strongly related to convex edges with $\gamma \ = \ \frac{3}{2}\pi$.



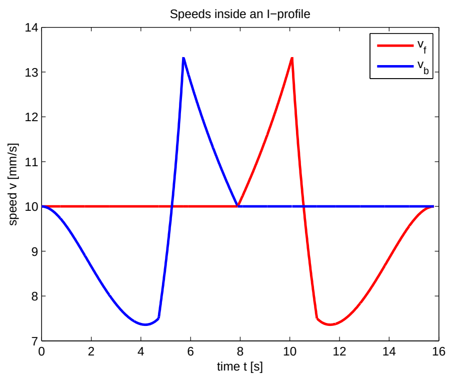


Figure 3.4: I-profile: situation overview and speed ratios.

Phases 1 and 6 are identical with the situation in Figure 3.2 (right), while phases 2 and 5 are comparable to the situation in Figure 3.2 (left) with $\gamma = \frac{3}{2}\pi$.
The new situations in phases 3 and 4 are discussed in detail in Appendix B.3. The speed ratios for all six phases are shown in Figure 3.4 (right).

3.2 Feedforward implementation details

The wheel speeds, obtained from the previous sub-chapters, are discretised by time t and implemented through classes written in Python. Depending on the obstacle, the feedforward class for concave edges, for convex edges or for I-profiles is activated. The obstacle angle γ, the robot absolute speed v_x and the time step dt are fed in from the main program. According to these values, the phase times are calculated for the convex and the I-profile case. Finally, the main function $step(self, currentv)$ sets the new speeds according to the previous calculations of this chapter. For a more precise control, a discretisation by the number of leading wheel revolutions n is implemented, too. The above equations can be adjusted easily as $v_i \cdot t = n_i \cdot 2r\pi$. Python code examples are shown in Appendix C.1.

3.3 Feedforward experiments

In order to test the quality of the feedforward approach, the following test plans are decisive. The primary goal is thereby to measure the remaining slip and the forces on the structure respectively. In addition to the verification of the control approach, the tests should as well reveal deeper insight into the slip and the torque behaviour inside obstacles.
Possible errors and factors of influence are tried to be anticipated in order to design the tests in a way that only one error occurs at the time. The experiments are conducted as reproducible as possible.
All the tests are completed with and without control in order to define the influence and appropriateness of the control strategies.
Further information is provided in Appendix D.

3.3.1 Test plans

The following abbreviations describe the testing results:

- S: test successful
- F: test failed
- +: criterion fulfilled
- -: criterion missed
- A: discussed in detail in Appendix D

Test scenarios for the general robot behaviour:

No.	**Issue**	**Scenario**	**Criterion**	**Result**
1	Speed output	Robot absolute speed v_x on an even surface	v_x agrees with the output speeds v_f, v_b.	S+
2	Torque output	Output torques on an even surface	Torques are equal for all wheels.	S+A
3	Torques and gravity	Output torques under various influences of gravity	Gravity influences torques.	S+A
4	Torques and curves	Output torques inside curves	Observation of the curve influence on torques	SA
5	Torques and obstacles	Output torques inside obstacles	Observation of the obstacle influence on torques	SA

Table 3.1: General testing.

Test scenarios for edges of different angles γ are shown in Table 3.2.

No.	**Issue**	**Scenario**	**Criterion**	**Result**
6	Control start	Drive towards an obstacle.	Control starts in time. $e \leq 0.1\,s$	S
7	Speed accuracy	Comparison of the predicted and the measured output speeds	$e \leq 5\,\%$	SA
8	Control stop	Time measurement when the robot fully overcame the obstacle	Complete ascent and control stop agree. $e \leq 5\,\%$ of the total overcoming time	S
9	Wheel slip	Comparison of the measured and the motor output distance	$e \leq 5\,\%$	S+A
10	Structural stress	Torque observation	Control reduces torques by 20 %.	S+A

Table 3.2: Obstacle testing.

According to the theoretical considerations and the objectives of the feedforward control approach, the following hypothesis is defined.

Feedforward hypothesis:
Feedforward speed control based on geometrical calculations reduces torque peaks inside obstacles.

3.3.2 Discussion of feedfoward results

In general, obstacles indeed have an influence on the torque behaviour. They cause high torque peaks inside concave and convex edges (cf. Appendix D.3). The reason for these peaks is the difference between the uncontrolled set speeds and the geometrically forced speeds, which is fought by the motor controllers (cf. Appendix D.6, Figure D.11). In addition to the above reason, the convex torques also occur because of the acceleration on the robot. This acceleration is possible as a result of the overlapping wheels. It is mainly caused by the magnetic forces, which make the robot flip over convex edges.
Regarding the torque behaviour, wheel slip should not be a problem with this set of experiments. It is theoretically impossible to build up torques if a wheel slips. Uncontrolled test runs prove that wheel slip is de facto negligible with a maximal amount of 0.26 % (cf. Appendix D.5).

Feedforward control is especially effective inside convex and concave edges. It leads to a reduction of peak torques of up to 40 %. The reduction is the higher, the more the obstacle angle γ differs from π. The feedforward hypothesis is hereby verified.
Slip measurements with a laser measurement device show the same result like for the general, uncontrolled test runs. This is very desirable as it proves the efficacy of the feedforward control approach. Moreover, driving precise trajectories is still possible.
The most important drawback of this control approach is the need to detect the obstacles and to estimate their angle γ. Furthermore, the control algorithm has to be started in time. The delay of an initialisation based on torque and speed behaviour (e.g., at the beginning of a concave edge, speeds decrease while torques increase or vice versa if the lever mechanism is mounted) turns out to be too extensive. Starting the control algorithm in time by using the human eye as a sensor leads to imprecision (cf. Table 3.2, No.6).
Finally, feedforward control for concave edges of $\gamma = \frac{\pi}{2}$ does not work very accurately. The reason is that the theoretical speeds become infinite, which is practically not feasible (cf. Figure 3.3 (left)). This results in a time shift between the calculated set speeds and the actual wheel speeds and further in an aberration in control stop (cf. Table 3.2, No.7 and 8). The effect gets worse with increasing convergence to $\gamma = \frac{\pi}{2}$. Also using symmetry within concave edges would increase the feedforward performance (cf. Appendix D.6).

Chapter 4

Feedback control approach based on torque comparisons

The goal of the feedback approach is to climb any kind of obstacles without knowing their geometry. Consequently, additional information is required to detect the need for different drives.

4.1 Concept evaluation

Five different concepts are discussed out of which the torque comparison control approach arises to be the most promising and is thus the one to be designed.

4.1.1 Stress detection with a resistive wire strain

The MagneBike [2] uses a resistive wire strain to observe any stress on its structure inside obstacles. Knowing the stress between the front and the rear wheels, the drives can be adjusted accordingly.
The resistive wire strain is a simple method to measure deformations. However, as it is imprecise, difficult to calibrate and sensitive to vibrations, it is generally not the best solution. Thinking of the massive stiffness and the short wheel axis distance of SIR, the success of a resistive wire strain seems highly questionable.

4.1.2 Deformation detection with an optical encoder

An optical encoder measures either the distance between the front and the rear wheels or the angle of the axis deformation. Knowing the change in distance or angle between the front and the rear allows to adjust the drives. However, the accuracy of optical encoders is most likely too limited to work on the very stiff and relatively small SIR.

4.1.3 Anti-slip control through absolute speed estimation

To deal with slip of cars, an estimation of the absolute car speed v_x is used and compared to the wheel speeds $\omega_i \cdot r$. If for any wheel $v_x - \omega_i \cdot r \neq 0$ (Figure 4.2), the controller will become active. Due to the strong magnetic wheels, no significant slip is expected for SIR. If the wheels slipped all the same, they would do it very suddenly and the slip was estimated to vanish again before it could be controlled. Thus, an anti-slip controller similar to the ones used for cars might not work precisely enough.

4.1.4 Detection of deviations from the motor characteristic

The used DC-motors have a linear dependency between the torque and the current (k_t) on the one hand, and between the drive and the voltage (k_{rpm}) on the other hand.
By defining a reference load case, deviations from the reference curve are easy to notice (Figure 4.1 (left)). If the front and the rear wheels are on different plates inside an obstacle, the load cases will differ. In order to control the wheels, the load cases for every possible relative wheel position is needed. On the one hand, this leads to countless load case experiments. On the other hand, the need to know the relative wheel position results in feedforward control.

4.1.5 Comparison of front and rear torques

Instead of the control of each wheel itself (cf. Chapter 4.1.4), the torque comparison control approach addresses the comparison of the torques of the front and the rear wheels (Figure 4.1 (right)). In situation 1, the robot moves on a flat plate f.l.t.r. All the wheels need the same rpm. In situation 2, the robot is inside an obstacle. The rpm of the front and the rear wheels should differ. However, at first the motor controllers (MCs) try to keep the rpm from situation 1. For one of the wheel pairs, the rpm from situation 1 is too low. For the other wheel pair it is too high. Consequently, one wheel pair pushes the other one and takes part of its load. Its torque increases while the other's decreases. A comparison of the torques allows to adjust the rpm accordingly.

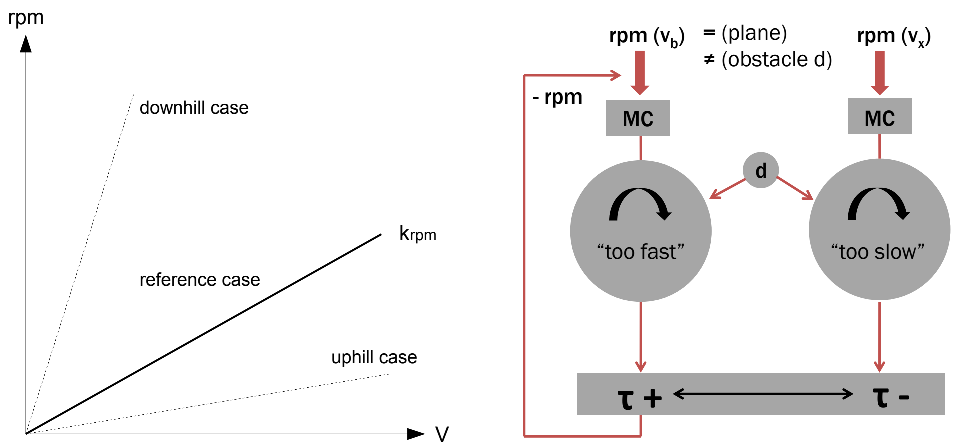


Figure 4.1: Voltage-drive dependency and torque control concept.

On the one hand, this approach might be imprecise due to the disturbing influence of the gravity force. The more negligible the robot mass can be considered, the better it works. Otherwise the load cases for the front and the rear wheels probably vary too much and strongly influence the torques themselves. On the other hand, it is an elegant, easy and new approach, which does not need any additional elements compared to the resistive wire strain or the optical encoder approach.

4.2 System overview

The system overview provides three important pieces of information.
Firstly, it describes SIR's geometry, which has a direct impact on the feedforward calculations.

Secondly, it presents an estimation of the gravity influence on wheel torques, which is of significance for the feedback control approach.
Thirdly, it visualises the software structure of Ship Inspection Robot and shows, where the control approaches are implemented.

4.2.1 Geometry of Ship Inspection Robot

A top view on SIR simplifies the geometry (Figure 4.2). The compact design and the low mass allow to position the centre of mass exactly in the middle without a significant loss of accuracy. The wheel configuration is therefore symmetrical.

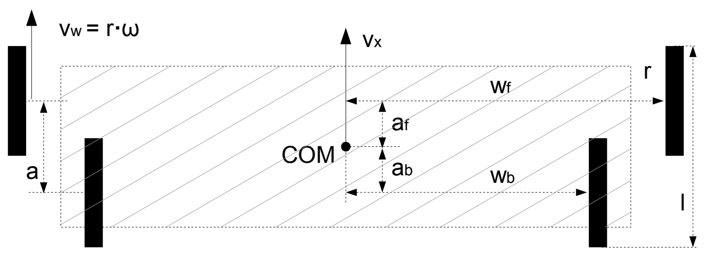


Figure 4.2: Robot model.

- Wheel radius: $r = 30\,mm$
- Absolute shaft centre distance: $a = 50\,mm$; $a_f = a_b = a/2$
- Robot length: $l = a + 2 \cdot r = 110\,mm$
- Front wheel distance to the COM: $w_f = 160\,mm$
- Rear wheel distance to the COM: $w_b = 110\,mm$
- Robot absolute speed: v_x
- Wheel speed: v_w

4.2.2 Estimation of the influence of gravity on the torque behaviour

A static force model is developed in order to gain a rough estimation of the influence of gravity on the torque comparison (cf. Chapter 4.1.5). Figure 4.3 provides an overview over the forces acting on each wheel.
In brief, the different geometrical load cases vary significantly (F_{up} versus F_{down}) although the magnetic force is more than 20 times higher than the gravity force (F_m versus F_g). This seems to be a problem for the torque control inside obstacles. However, as the robot is one stiff body, the additional load generated by the gravity force will always be shared between the front and the rear. In other words, the downhill-slope force F_d induces an acceleration on the whole robot, including all the four wheels.
An accurate dynamic model of the robot including the magnetic forces, the wheel slip and the geometrical constraints of the obstacles would go beyond the scope of this thesis. Instead, extensive testing clarifies the torque behaviour under the influence of gravity (cf. Chapter 4.4.1).

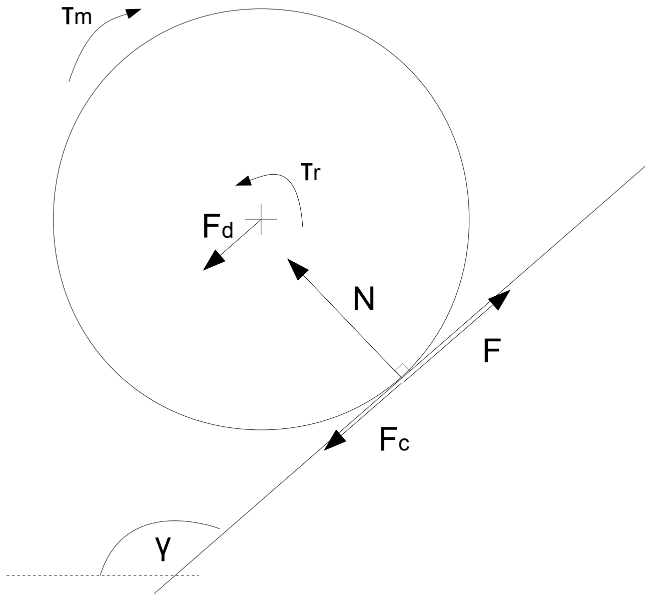


Figure 4.3: Forces acting on each wheel.

- Normal force: $N = F_m + F_g \cdot cos(\pi - \gamma)$ where:
  - Magnetic force: $F_m \approx 400\,N$
  - Gravitational force: $F_g = \frac{m_{robot} \cdot g}{4} \approx 18.4\,N$
    with: robot mass: $m_{robot} \approx 7.5\,kg$
- Active force: $F = \frac{\tau_m}{r} \leq 133\,N$
- Counter-force: $F_c = N \cdot \mu \leq 42\,N$ where: $\mu \leq 0.1$ (steel-steel)
- Motor torque: $\tau_m \leq 4\,Nm$
- Rolling resistant moment: $\tau_r = F \cdot r$
- Downhill-slope force: $F_d = F_g \cdot sin(\pi - \gamma)$

- Vertical uphill opposing force (statical balance of forces):
  $F_{up} = 4 \cdot (F_c + F_g) \approx 235\,N$
- Vertical downhill opposing force (statical balance of forces):
  $F_{down} = 4 \cdot (F_c - F_g) \approx 85\,N$

4.2.3 Software structure of Ship Inspection Robot

Figure 4.4 shows the software structure of SIR, which has four motors (M). The motion starts with a user who pushes a button on a gamepad. The readout pieces of information (angular and linear velocity) are converted to four rpm inside the planar control block.
Subsequently, the cascaded control blocks adjust the four rpm if required. The need for an adjustment can arise either through an external feedforward algorithm start

command or through a series of different torque feedbacks of the four motors, which activates the feedback control approach.
The adjusted rpm pass through the drivetrain manager, where they are scaled to motor speeds.
Finally, the four speed commands go to the four motors, which each have a motor controller for speed control from Alstom Inspection Robotics (AIR) on top.

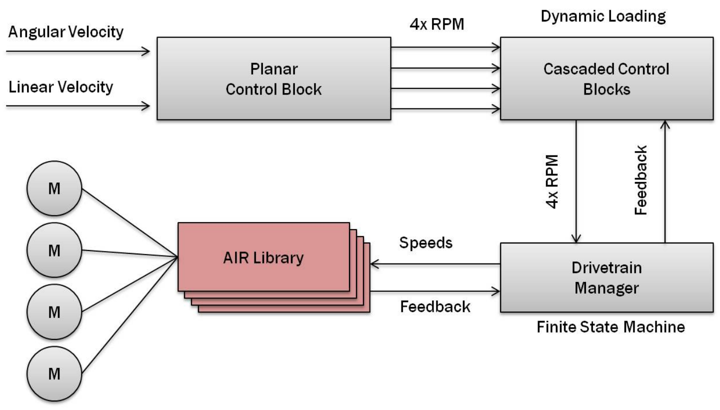


Figure 4.4: Software structure of SIR.

4.3 Feedback implementation details

The feedback concept is again implemented through a Python class. Compared to the feedforward classes, this class does not need to be activated. While running all the time, it compares the torques and starts to control them automatically as soon as the front and the rear wheel torques become different.
The control is proportional. The higher the torque difference, the stronger the speed adjustments become.
The complete Python code is shown in Appendix C.2.

4.4 Feedback experiments

The feedback testing undergoes the same procedure like the feedforward testing (cf. Chapter 3.3.1). The hypothesis to be proved for feedback control is the following.

Feedback hypothesis:
Feedback speed control based on a wheel torque comparison reduces torque peaks inside obstacles and leads to wheel speeds according to the feedforward calculations.

4.4.1 Discussion of feedback results

Fortunately, the gravity influence on the torque behaviour is up to five times smaller than the obstacle influence (cf. Figures D.1, D.2 and D.5). A useful feedback control is thus possible. Moreover, Figure D.11 confirms the torque comparison concept (cf. 4.1.5).
In brief, feedback control is particularly effective inside concave edges, where it causes the same torque reduction like the feedforward approach of maximum 40 %. Slip is negligible, too. Therefore, the feedback hypothesis is confirmed. The big advantage of the feedback control approach is that obstacles do not have to be

identified.

However, the feedback control inside convex edges is more complex than anticipated. The overlapping wheel configuration of SIR and the highly magnetic wheels cause a flipping mechanism whenever the robot moves from one to another plate inside convex edges. This flipping results in an acceleration, which makes the torques increase (cf. Appendix D.3.3). The effect gets worse with increasing convergence to $\gamma = \frac{3}{2}\pi$.
There are two possibilities to counter this problem. First, a different wheel configuration (higher absolute shaft centre distance) would make the flipping geometrically impossible. Second, a faster control algorithm would increase the impact on the torques during the acceleration phase. The feedforward speed control approach works inside convex edges because speeds are directly controllable. To fasten the feedback control, either motors with directly controllable torques or an on-board computer for control issues would have to be installed.

Chapter 5

Conclusion

Obstacle climbing can end in various disadvantageous impacts on the climbing vehicles (see Figure 5.1). In the case of Ship Inspection Robot, high torque peaks on the drivetrain turn out to be the problem. The conclusion aims to discuss the designed solutions to counter theses torque peaks.

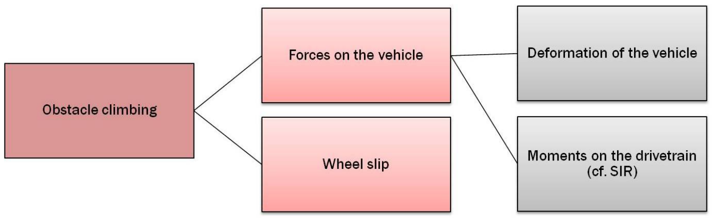

Obstacle climbing
Forces on the vehicle
Wheel slip
Deformation of the vehicle
Moments on the drivetrain (cf. SIR)

Figure 5.1: Possible influences of obstacle climbing.

5.1 Comparison between feedforward and feedback solutions

At first sight, feedforward control seems to be more appropriate than the feedback solution. It works properly on almost every kind of obstacle. However, feedforward control has a massive drawback: Obstacles need to be detected and characterised. In more detail, the obstacle angle γ has to be estimated and the control algorithm has to be started just in time. On top of that, the calculations and the implementation of feedforward control are much more complicated than the ones of the feedback control algorithm. The former requires numerous parameter adaptions as soon as the robot design changes.

Besides, running feedforward and feedback control at the same time does not improve the results.

In conclusion, the feedback solution is the one which has fair chances to be applied on future obstacle climbing robots. Moreover, it is not only appropriate for torque control on climbing robots, but also for anti-slip control on electrically driven vehicles (Table 5.1).

What needs to be done to reach marketability is described in Chapter 5.3.

	✓	ϟ
Feedforward	Any obstacles possible	Obstacle identification required
Feedback	Suitable for anti-slip control	Flipping mechanism inside convex edges

Table 5.1: Control comparison.

5.2 Limitations of the designed controllers

The controllers can reduce torques or wheel slip and therefore enhance the drive performance on uneven surfaces. However, they cannot empower an unsuitably designed robot to climb obstacles. A realistic evaluation of their contribution to the drive performance leads to the comparison with Anti-lock Brake Systems (ABS) and the Electronic Stability Control (ESC) for cars: These systems help to stabilise cars within narrow boundaries. However, it is possible to drive cars in a safe way without ABS and ESP and to drive unsafely despite them.

In addition to the discussed weaknesses and troubles with the control approaches, the various factors of influence complicate the design of the controllers. Generally, the torque behaviour is influenced by the wheel alignment precision and the precision in approaching the obstacles straight as well as the magnetic forces of the wheels.
With feedforward control, the wheel radius, the algorithm starting time and the motor acceleration capacity disturb the precise speed control.
The flipping mechanism finally has a negative impact on the feedback control inside convex edges.
These factors would have to be reduced in order to shift the limitations of the control approaches.

5.3 Outlook for future improvements

In general, investing more time in a better understanding of the system as well as the torque and the wheel slip behaviour would deliver deeper insight into the factors of influence and their connections. Complex system modelling would surely provide additional knowledge.

5.3.1 Feedforward recommendations

If the feedforward controller wants to be used in a real operating scenario, the obstacle identification would be the first thing to be optimised. Ideally, an autonomous obstacle detection, obstacle angle estimation and control start process should be designed. With respect to Ship Inspection Robot, its infrared sensors as well as the camera could be helpful in doing so. The usage of the IMU seems rather disadvantageous as it leads to a delay between the obstacle detection and the start of the control algorithm.

The problem of infinite theoretical speeds inside concave edges should be solved by using symmetry. The theoretical solution is presented in Appendix D.6. What remains to be done is the implementation and testing on the robot.

For a more precise speed control inside concave edges, the detaching mechanism, which Ship Inspection Robot uses for the transition from one to another plate, would be needed to be taken into account. This lever decreases the magnetic forces on one side of the concave edge in order to allow the continuation of the robot's motion. It would add two more phases to the concave obstacle climbing.

The equations for feedforward control are discretised by time and by wheel revolutions that are fed in by the encoder positions of the motors. The scarce time resources do not allow to test the wheel revolution approach. It might be advantageous as the dependency of the time disappears when using it.

I-profiles are theoretically calculated but not yet tested. It is an easy step to prove the usefulness of the I-profile control algorithm as soon as SIR is able to drive them reliably.

Lastly, the data transmission to the robot and the acceleration capacity of the motors should be enhanced up to a level where it is feasible to follow quick and high speed changes.

5.3.2 Feedback recommendations

The feedback control approach is convincing due to its simplicity and reliability. A shorter delay between the detection of torque aberrations and the according corrective speed commands would be the most obvious improvement. This is especially crucial while driving wirelessly, when the data transmission is the slowest. An on-board computer for control issues would surely improve the current situation.

Further experiments could show on which wheels it makes most sense to place the lead and which wheels should be controlled respectively. So far, the lead is always on the front wheels and does not change while driving backwards. This leads to speed ratios inside obstacles like in Figure D.9 (right).
However, in Appendix D.6 it is estimated that symmetrical speed ratios are advantageous for control approaches. Therefore, an adjustment of the feedback control algorithm towards symmetrical speed outcomes (cf. feedforward speed ratios in Figure D.8 (right)) would probably improve the feedback performance and thus decrease the influence of the present flipping mechanism.

5.4 Possible applications

Obstacle Climbing Control can be applied to any vehicle with more than two electric motors or with installed wheel torque measurement devices. The rougher the ground, the more useful it becomes.
Within this thesis, torque peaks are the problem which is fought by Obstacle Climbing Control. Especially the MagneBike could benefit a lot from Obstacle Climbing Control as it has massive trouble with deformation due to its flexible design.
The problem to solve could just as well be wheel slip. The effect of wheel slip are often decreasing torques. Theoretically, they could be detected and countered with the proposed feedback algorithm of Obstacle Climbing Control. Implemented on electric cars, Obstacle Climbing Control might be a future alternative to Anti-lock Brake Systems (ABS) or to the Electronic Stability Control (ESC).

5.5 Relevance of Obstacle Climbing Control

Obstacle Climbing Control lowers the torques on the drivetrain significantly and therefore enhances the lifetime of obstacle climbing robots. For the very first application, which is the stiff and compact Ship Inspection Robot, it is not absolutely necessary though beneficial. The big advantage of Obstacle Climbing Control over current solutions is that it works without any additional measurement devices.

Appendix A

Testing environment

A.1 Stability analysis

The framework and the testing ground are stable. The critical parts are the mountings (red) on the cross profiles, which connect the framework and the testing ground. The connecting item slot nuts can deal with a maximum force $F_{max} = 4000\,N$. In order to minimise the forces on the mountings, the testing ground is mounted along its centroidal axis. The remaining load is due to the robot mass $m_{robot} = 7.5\,kg$.

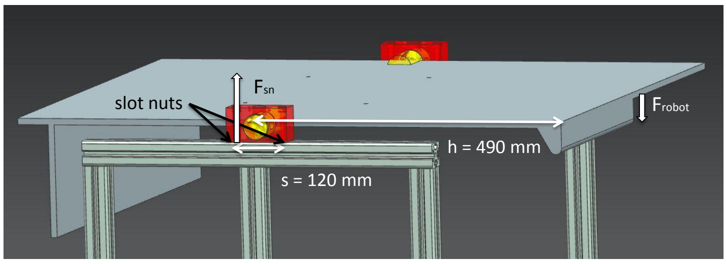


Figure A.1: Forces on the mountings.

$$F_{robot} = m_{robot} * g = 73.6\,N$$

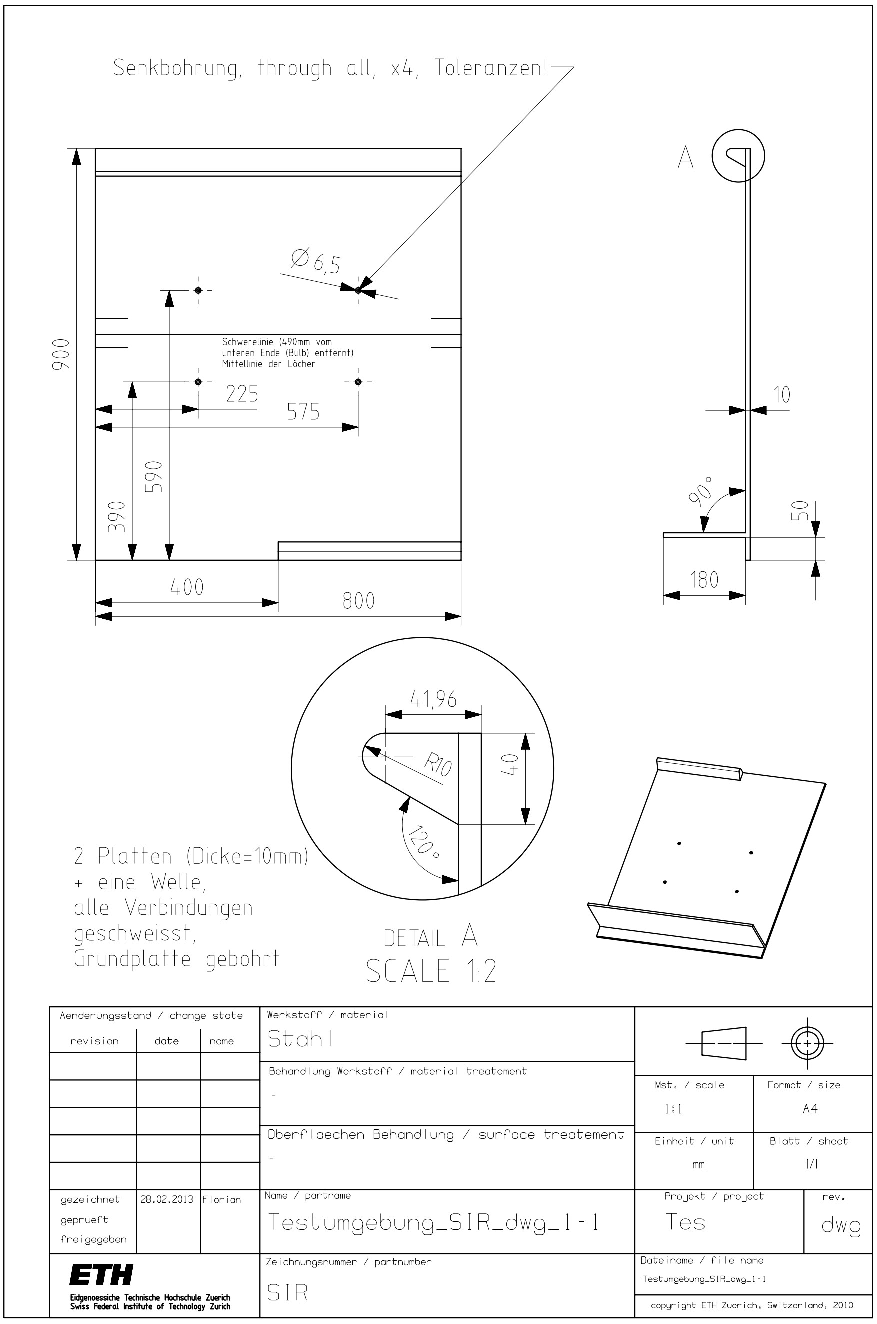
The robot force F_{robot} has to be divided by two as there are two mountings, which share the force.

$$s \cdot F_{sn} = (h - \frac{s}{2}) \cdot \frac{F_{robot}}{2}$$

$$F_{sn} = \frac{h - \frac{s}{2}}{s} \cdot \frac{F_{robot}}{2} = 131.8\,N \tag{A.1}$$

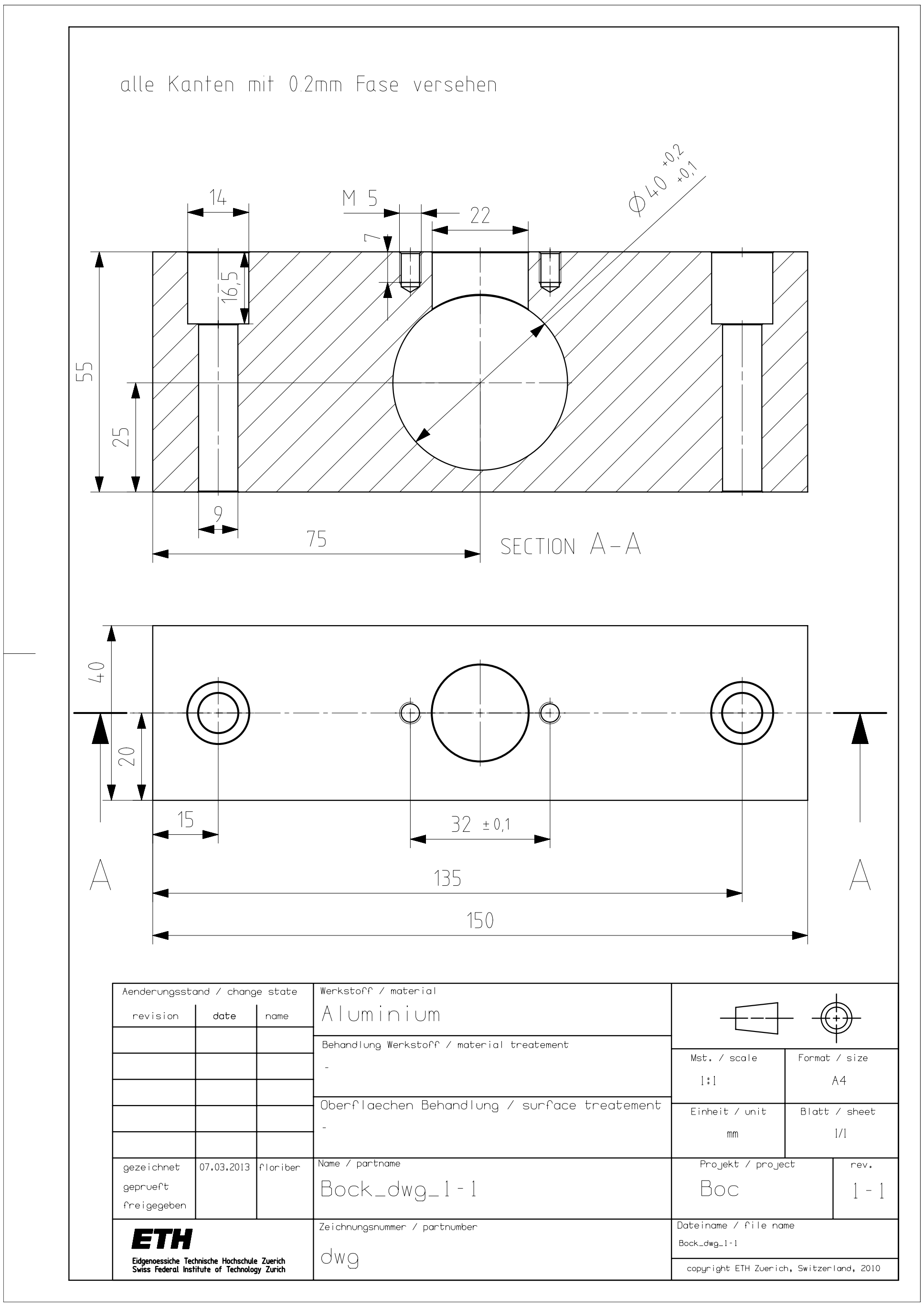
Therefore, $F_{sn} < F_{max}$. The slot nuts are strong enough.

A.2 Technical drawings


Senkbohrung, through all, x4, Toleranzen!
⌀6,5
Schwerelinie (490mm vom unteren Ende (Bulb) entfernt) Mittellinie der Löcher
900
590
390
225
575
400
800
A
10
90°
50
180
41,96
R10
40
120°
2 Platten (Dicke=10mm) + eine Welle, alle Verbindungen geschweisst, Grundplatte gebohrt
DETAIL A
SCALE 1:2

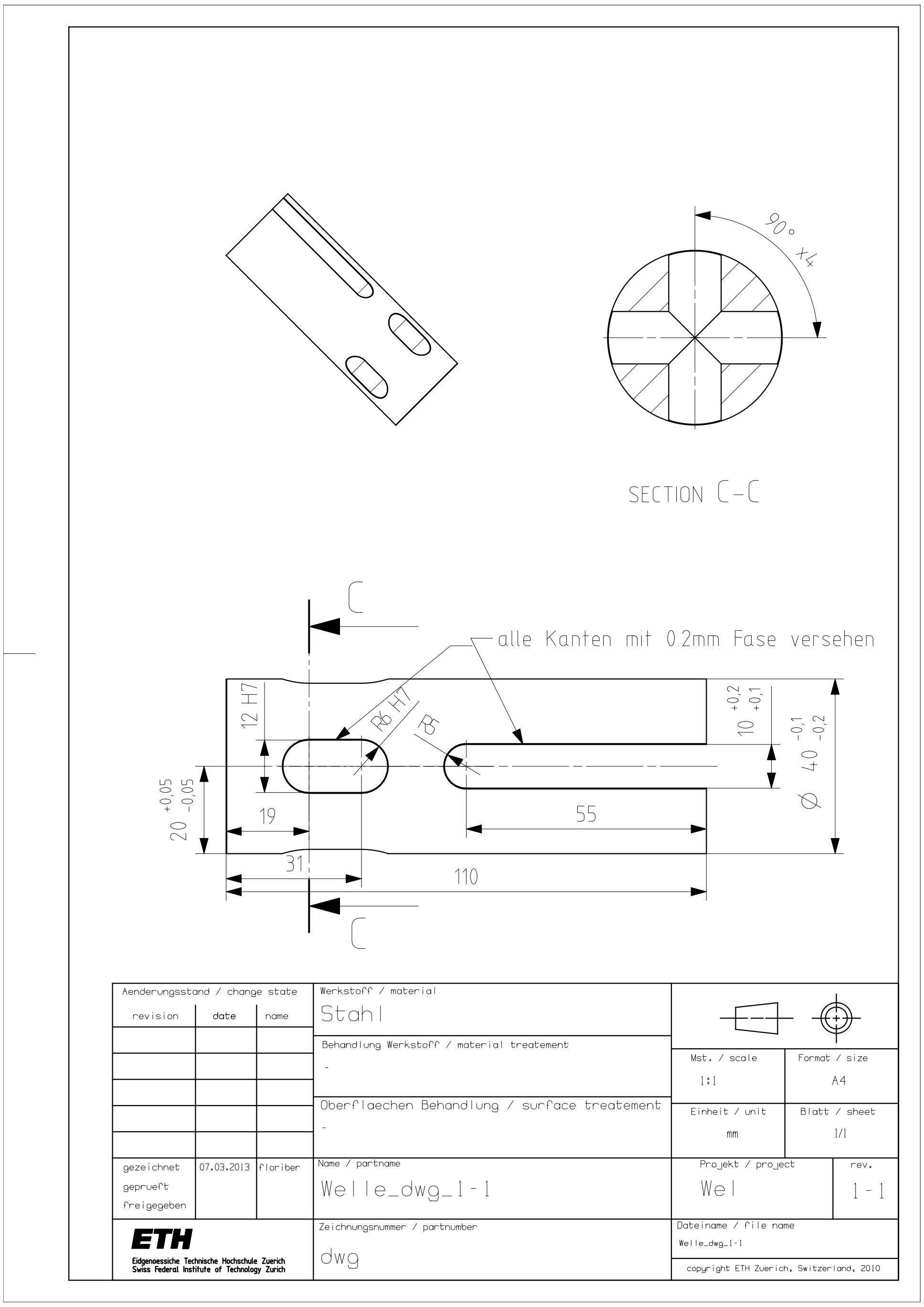
Aenderungsstand / change state			Werkstoff / material: Stahl		
revision	date	name	Behandlung Werkstoff / material treatement: -	Mst. / scale: 1:1	Format / size: A4
			Oberflaechen Behandlung / surface treatement: -	Einheit / unit: mm	Blatt / sheet: 1/1
gezeichnet	28.02.2013	Florian	Name / partname: Testumgebung_SIR_dwg_1-1	Projekt / project: Tes	rev.: dwg
geprueft					
freigegeben					
ETH Eidgenoessische Technische Hochschule Zuerich Swiss Federal Institute of Technology Zurich			Zeichnungsnummer / partnumber: SIR	Dateiname / file name: Testumgebung_SIR_dwg_1-1	
				copyright ETH Zuerich, Switzerland, 2010	

Figure A.2: Base plate.



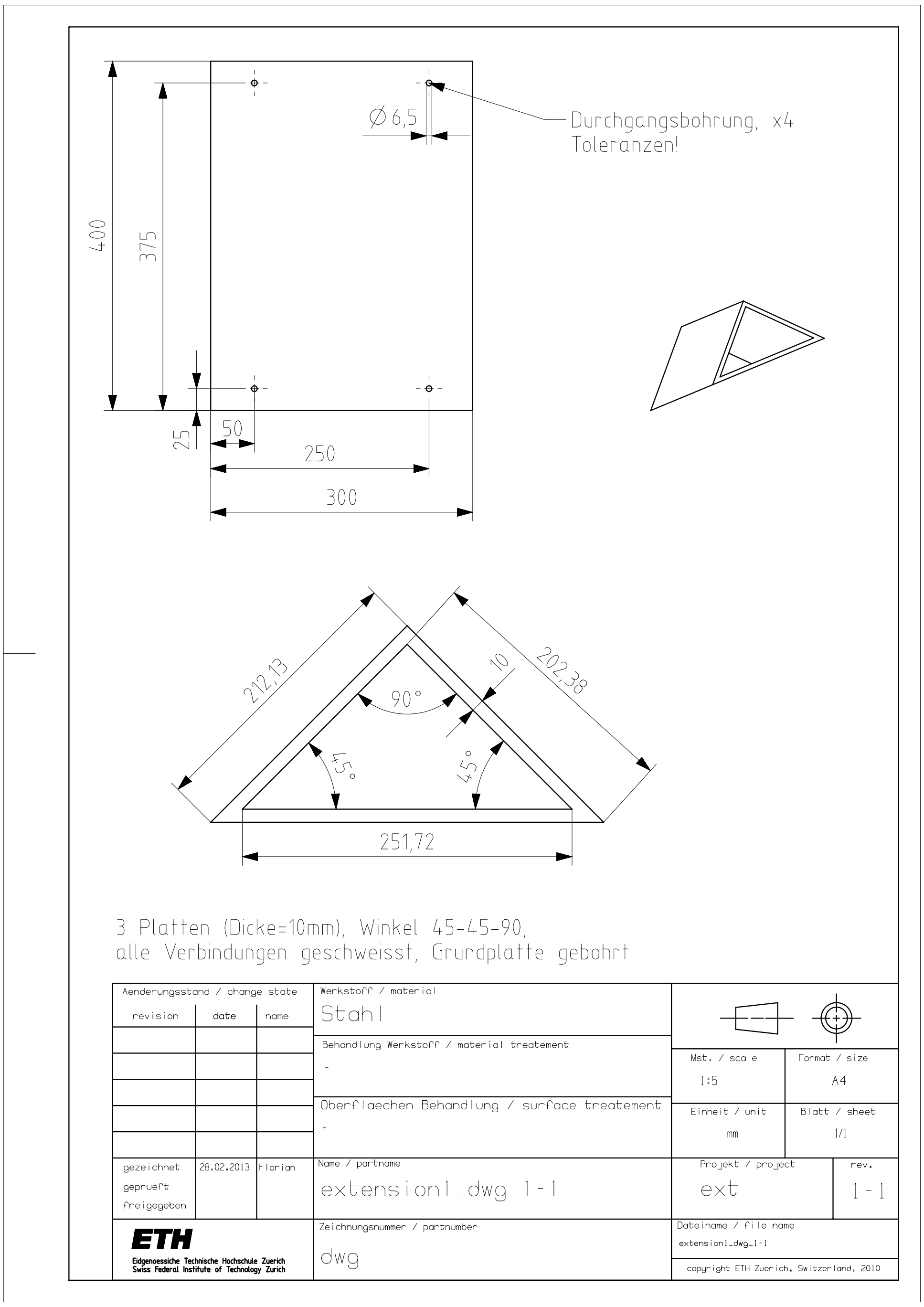
Aenderungsstand / change state			Werkstoff / material	
revision	date	name	Aluminium	
			Behandlung Werkstoff / material treatement -	Mst. / scale 1:1 · Format / size A4
			Oberflaechen Behandlung / surface treatement -	Einheit / unit mm · Blatt / sheet 1/1
gezeichnet geprueft freigegeben	07.03.2013	floriber	Name / partname Bock_dwg_1-1	Projekt / project Boc · rev. 1-1
ETH Eidgenoessiche Technische Hochschule Zuerich Swiss Federal Institute of Technology Zurich			Zeichnungsnummer / partnumber dwg	Dateiname / file name Bock_dwg_1-1 copyright ETH Zuerich, Switzerland, 2010

Figure A.3: Mounting block.



Aenderungsstand / change state			Werkstoff / material Stahl		
revision	date	name	Behandlung Werkstoff / material treatement -	Mst. / scale 1:1	Format / size A4
			Oberflaechen Behandlung / surface treatement -	Einheit / unit mm	Blatt / sheet 1/1
gezeichnet geprueft freigegeben	07.03.2013	floriber	Name / partname Welle_dwg_1-1	Projekt / project Wel	rev. 1-1
ETH Eidgenoessiche Technische Hochschule Zuerich Swiss Federal Institute of Technology Zurich			Zeichnungsnummer / partnumber dwg	Dateiname / file name Welle_dwg_1-1 copyright ETH Zuerich, Switzerland, 2010	

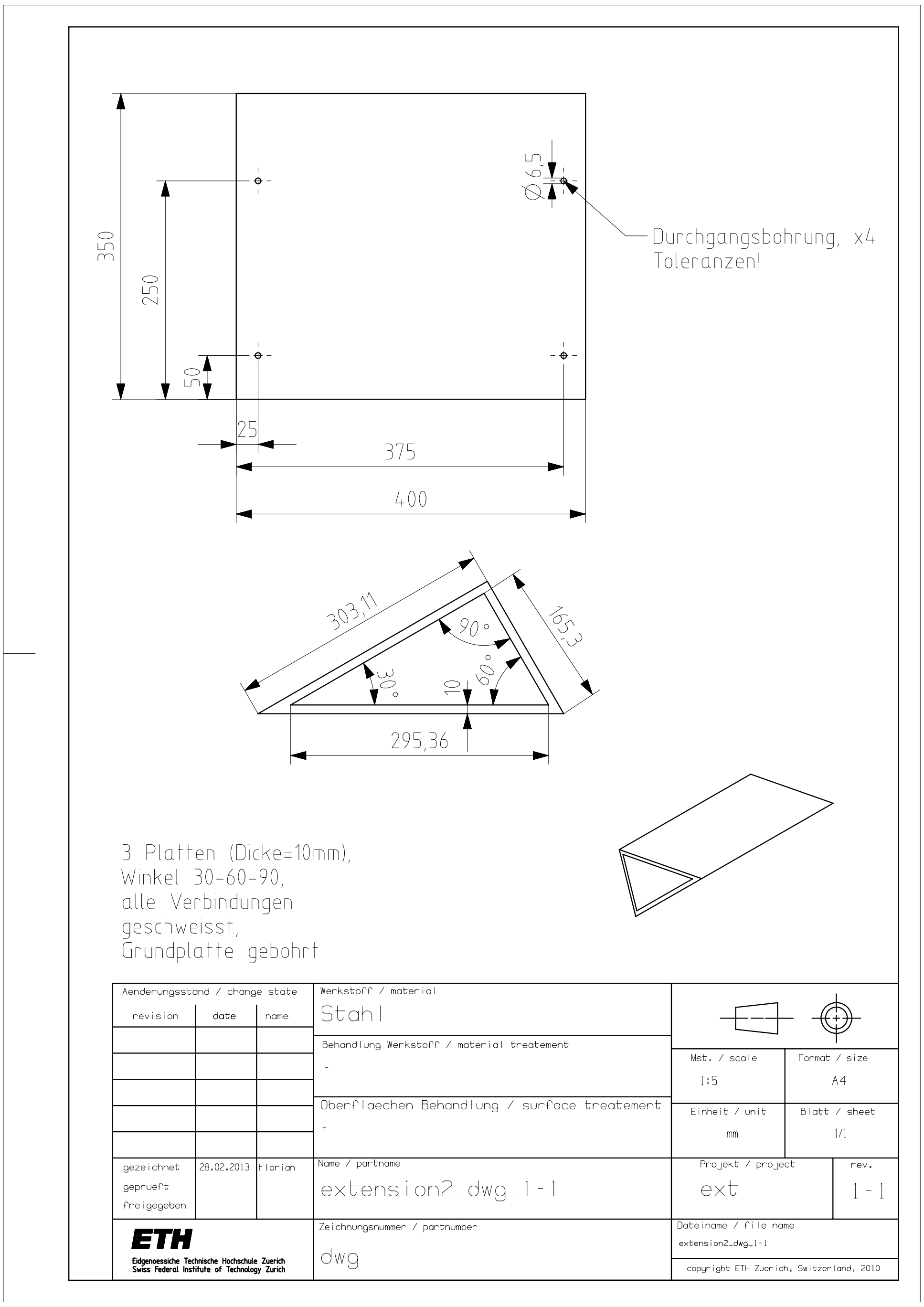
Figure A.4: Mounting axle.



3 Platten (Dicke=10mm), Winkel 45-45-90,
alle Verbindungen geschweisst, Grundplatte gebohrt

Aenderungsstand / change state			Werkstoff / material		
revision	date	name	Stahl		
			Behandlung Werkstoff / material treatement	Mst. / scale	Format / size
			-	1:5	A4
			Oberflaechen Behandlung / surface treatement	Einheit / unit	Blatt / sheet
			-	mm	1/1
gezeichnet	28.02.2013	Florian	Name / partname	Projekt / project	rev.
geprueft			extension1_dwg_1-1	ext	1-1
freigegeben					
ETH Eidgenoessische Technische Hochschule Zuerich Swiss Federal Institute of Technology Zurich			Zeichnungsnummer / partnumber	Dateiname / file name	
			dwg	extension1_dwg_1-1	
				copyright ETH Zuerich, Switzerland, 2010	

Figure A.5: 45°-45°-90° obstacle.



3 Platten (Dicke=10mm),
Winkel 30-60-90,
alle Verbindungen
geschweisst,
Grundplatte gebohrt

Aenderungsstand / change state			Werkstoff / material		
revision	date	name	Stahl		
			Behandlung Werkstoff / material treatement: -	Mst. / scale: 1:5	Format / size: A4
			Oberflaechen Behandlung / surface treatement: -	Einheit / unit: mm	Blatt / sheet: 1/1
gezeichnet / geprueft / freigegeben	28.02.2013	Florian	Name / partname: extension2_dwg_1-1	Projekt / project: ext	rev.: 1-1
ETH Eidgenoessiche Technische Hochschule Zuerich, Swiss Federal Institute of Technology Zurich			Zeichnungsnummer / partnumber: dwg	Dateiname / file name: extension2_dwg_1-1	

Figure A.6: 30°-60°-90° obstacle.

Appendix B

Feedforward calculations

While rotating a stiff body, all its parts have to move with the same angular speed $\omega = \frac{v_i}{r_i}$. The knowledge of the instantaneous centre of rotation and the distances r_i to body parts allows the calculation of the speed ratios between body parts. For the discussed robot and the environment, the following parameters are given (cf. Figure 4.2):

- Obstacle angle: γ
- Wheel radius: $r = 30\,mm$
- Absolute shaft centre distance: $a = 50\,mm$
- Robot absolute speed: v_x

The unknown parameters are:

- Front wheel distance to the ICR: f
- Rear wheel distance to the ICR: b
- Adjusted wheel speeds: v_f or v_b, depending on the situation

Once f and b are defined through geometrical calculations, the speed ratio is simply

$$v_b \cdot f = v_f \cdot b \tag{B.1}$$

The horizontally projected shaft centre distance x and the vertically projected shaft centre distance y are introduced as auxiliary variables. The covered distance of the wheels is $v_i \cdot t = n_i \cdot 2r\pi$. This allows to discretise the calculations either by time t or by wheel revolutions n. The control is always about a speed ratio. Therefore, it does not matter whether speeds, drives or rpm are compared between the front and the rear and which unit they refer to.

B.1 Concave edges

The concave situation is shown in Figure 3.1.

$$f = \frac{x}{sin(\pi - \gamma)}$$

$$b = \frac{x}{tan(\pi - \gamma)} + y$$

where:

$$x = \sqrt{a^2 - y^2}$$

$$y = a_{pre} \cdot sin(\pi - \gamma) = v_f \cdot \ t \cdot sin(\pi - \gamma)$$

Hence:

$$\frac{b}{f} = cos(\pi - \gamma) + \frac{y}{x} \cdot sin(\pi - \gamma) = -cos\gamma + \frac{v_f \cdot \ t \cdot sin^2\gamma}{\sqrt{a^2 - (v_f \cdot \ t \cdot sin\gamma)^2}} \tag{B.2}$$

B.2 Convex edges

The following equations are considering the front wheels as the leaders, which means that $v_f = v_x$. The general convex situation (phases 2 and 3), where all the wheels are outside the sector between point 1 and 2, is similar to the concave case (Figure 3.2 (left)):

$$f = \frac{x}{sin(\gamma - \pi)}$$

$$b = \frac{x}{tan(\gamma - \pi)} + y$$

where:

$$x = \sqrt{a^2 - y^2}$$

$$y = a_{pre} \cdot sin(\pi - \gamma) + s = v_f \cdot \ t \cdot sin(\gamma - \pi) + s$$

And:

$$s = cos(\pi - \frac{\gamma}{2}) \cdot l$$

where:

$$l = 2r \cdot sin(\frac{\gamma - \pi}{2})$$

thus:

$$s = 2r \cdot cos^2(\frac{\gamma}{2})$$

Finally for phase 2:

$$\frac{b}{f} = cos(\gamma - \pi) + \frac{y}{x} \cdot sin(\gamma - \pi) = -cos\gamma - sin\gamma \cdot \frac{-v_f \cdot \ (t - t_1) \cdot sin\gamma + 2r \cdot cos^2(\frac{\gamma}{2})}{\sqrt{a^2 - (-v_f \cdot \ (t - t_1) \cdot sin\gamma + 2r \cdot cos^2(\frac{\gamma}{2}))^2}} \tag{B.3}$$

In order to change the lead from the front to the rear in time, the symmetry point has to be known. The covered front wheel distance from point 2 to the symmetry point is a_{sym}. At the symmetry point, f is equal to b:

$$f = b \Leftrightarrow \frac{x}{sin(\gamma - \pi)} = \frac{x}{tan(\gamma - \pi)} + y$$

Therefore:

$$y = \frac{a \cdot \sqrt{2 \cdot (cos\gamma + 1)}}{2}$$

and also:

$$y = -a_{sym} \cdot sin\gamma + s$$

Finally:

$$a_{sym} = \frac{-y + s}{sin\gamma} = \frac{\frac{-a \cdot \sqrt{2 \cdot (cos\gamma+1)}}{2} + s}{sin\gamma} \tag{B.4}$$

If one wheel pair is exactly on the sector between point 1 and 2 (phases 1 and 4), a new situation arises (Figure 3.2 (right)):

$$f = \frac{x}{sin(\alpha)}$$

$$b = \frac{x}{tan(\alpha)} + y$$

Where:

$$x = \sqrt{a^2 - y^2}$$

$$y = cos(\frac{\pi - \alpha}{2}) \cdot l$$

and:

$$l = 2r \cdot sin\frac{\alpha}{2}$$

$$\alpha = \frac{v_f \cdot \ t}{r}$$

Therefore for phase 1:

$$\frac{b}{f} = cos\alpha + \frac{y}{x} \cdot sin\alpha = cos(\frac{v_f \cdot \ t}{r}) + sin(\frac{v_f \cdot \ t}{r}) \cdot \frac{2r \cdot sin^2(\frac{v_f \cdot \ t}{2r})}{\sqrt{a^2 - \left(2r \cdot sin^2(\frac{v_f \cdot \ t}{2r})\right)^2}} \tag{B.5}$$

Lastly, the phase times are determined in order to change the calculations for the speed ratios in time:

$$\begin{aligned} t_1 &= r \cdot \frac{\gamma - \pi}{v_x} \\ t_2 &= t_1 + \frac{a_{sym}}{v_x} \\ t_3 &= t_1 + 2 \cdot \frac{a_{sym}}{v_x} \\ t_4 &= 2 \cdot t_2 \end{aligned} \tag{B.6}$$

During phase 1 and 2, the above equations are in use. After "halftime" and the change of the lead, v_f and v_b are exchanged and the time is counted backwards. This leads to the following equation for phase 3 (cf. equation B.3):

$$\frac{f}{b} = -cos\gamma - sin\gamma \cdot \frac{-v_b \cdot \ (t_3 - t) \cdot sin\gamma + 2r \cdot cos^2(\frac{\gamma}{2})}{\sqrt{a^2 - (-v_b \cdot \ (t_3 - t) \cdot sin\gamma + 2r \cdot cos^2(\frac{\gamma}{2}))^2}} \tag{B.7}$$

And finally the equation for phase 4 (cf. equation B.5):

$$\frac{f}{b} = cos(\frac{v_b \cdot \ (t_4 - t)}{r}) + sin(\frac{v_b \cdot \ (t_4 - t)}{r}) \cdot \frac{2r \cdot sin^2(\frac{v_b \cdot \ (t_4 - t)}{2r})}{\sqrt{a^2 - \left(2r \cdot sin^2(\frac{v_b \cdot \ (t_4 - t)}{2r})\right)^2}} \tag{B.8}$$

B.3 I-profiles

The I-profile situations 1 and 6 agree with the convex edge phases 1 and 4. Therefore, their equations are similar (cf. equations B.5 and B.8).

Situation 1:

$$\frac{b}{f} = cos(\frac{v_f \cdot \ t}{r}) + sin(\frac{v_f \cdot \ t}{r}) \cdot \frac{2r \cdot sin^2(\frac{v_f \cdot \ t}{2r})}{\sqrt{a^2 - \left(2r \cdot sin^2(\frac{v_f \cdot \ t}{2r})\right)^2}} \tag{B.9}$$

Situation 6:

$$\frac{b}{f} = cos(\frac{v_b \cdot \ (t_6 - t)}{r}) + sin(\frac{v_b \cdot \ (t_6 - t)}{r}) \cdot \frac{2r \cdot sin^2(\frac{v_b \cdot \ (t_6 - t)}{2r})}{\sqrt{a^2 - \left(2r \cdot sin^2(\frac{v_b \cdot \ (t_6 - t)}{2r})\right)^2}} \tag{B.10}$$

Moreover, situations 2 and 5 on I-profiles accord with the convex edge phases 2 and 3 (cf. equations B.3 and B.7).

Situation 2:

$$\frac{b}{f} = -cos\gamma - sin\gamma \cdot \frac{-v_f \cdot \ (t - t_1) \cdot sin\gamma + 2r \cdot cos^2(\frac{\gamma}{2})}{\sqrt{a^2 - (-v_f \cdot \ (t - t_1) \cdot sin\gamma + 2r \cdot cos^2(\frac{\gamma}{2}))^2}} \tag{B.11}$$

Situation 5:

$$\frac{b}{f} = -cos\gamma - sin\gamma \cdot \frac{-v_b \cdot \ (t_5 - t) \cdot sin\gamma + 2r \cdot cos^2(\frac{\gamma}{2})}{\sqrt{a^2 - (-v_b \cdot \ (t_5 - t) \cdot sin\gamma + 2r \cdot cos^2(\frac{\gamma}{2}))^2}} \tag{B.12}$$

Situation 3 on I-profiles, where the front and the rear are on circular paths, is different to all the previously discussed situations (Figure B.1). The situation starts in point 2f and ends in point 3 (symmetry point). The leading front wheels are the origin of a moving coordinate system, labelled with p_f.
Unfortunately, the complexity of the equations does not allow to insert all the variables into the final solution.

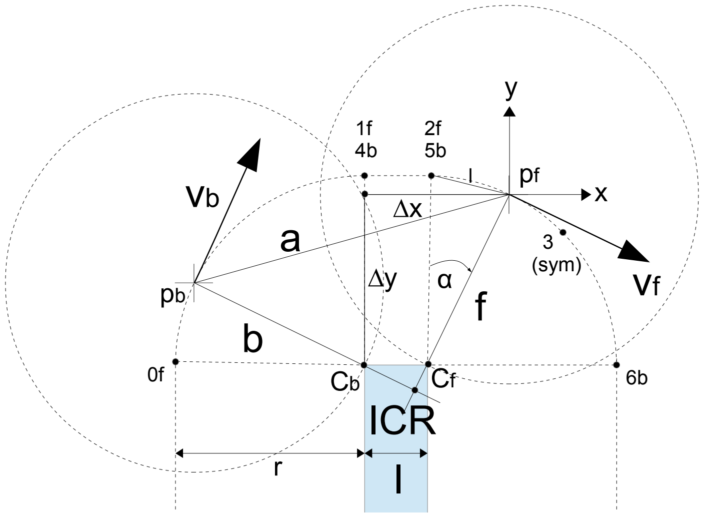


Figure B.1: I-profile: climbing situation 3.

The running angle α and the chord length l are

$$\alpha = \frac{v_f}{r} \cdot (t - t_2)$$

$$l = 2r \cdot sin(\frac{\alpha}{2})$$

Therefore, the distances from point p_f to point C_b are Δx and Δy:

$$\Delta x = I + l \cdot cos\frac{\alpha}{2}$$

$$\Delta y = r - l \cdot sin\frac{\alpha}{2}$$

The rear wheel position p_b is evaluated by intersecting two circles.
Circle 1 is around the front wheels p_f (origin) with diameter a (absolute shaft centre distance):

$$x^2 + y^2 = a^2$$

Circle 2 is around point C_b with diameter r (wheel radius):

$$(x + \Delta x)^2 + (y + \Delta y)^2 = r^2$$

The equalisation of circle 1 and circle 2 delivers the position of the rear wheel centre p_b:

$$x_{pb} = -\frac{\Delta y \cdot \sqrt{-r^4 + 2 \cdot (a^2 + \Delta x^2 + \Delta y^2) \cdot r^2 - a^4 + 2 \cdot a^2 \cdot (\Delta x^2 + \Delta y^2) - \Delta x^4 - 2 \cdot \Delta x^2 \cdot \Delta y^2 - \Delta y^4} - \Delta x \cdot (r^2 - a^2 - \Delta x^2 - \Delta y^2)}{2 \cdot (\Delta x^2 + \Delta y^2)}$$

$$y_{pb} = \frac{\Delta x \cdot \sqrt{-r^4 + 2 \cdot (a^2 + \Delta x^2 + \Delta y^2) \cdot r^2 - a^4 + 2 \cdot a^2 \cdot (\Delta x^2 + \Delta y^2) - \Delta x^4 - 2 \cdot \Delta x^2 \cdot \Delta y^2 - \Delta y^4} + \Delta y \cdot (r^2 - a^2 - \Delta x^2 - \Delta y^2)}{2 \cdot (\Delta x^2 + \Delta y^2)}$$

Now that the front p_f and the rear p_b are mathematically located, the ICR can be found by intersecting the two half-lines f and b (linear equation: $y = mx + q$).
Half-line f from the front wheel centre p_f through C_f:

$$m_f = \frac{1}{tan(\alpha)}, \; q_f = 0$$

Half-line b from the rear wheel centre p_b through C_b:

$$m_b = \frac{-\Delta y - y_{pb}}{-\Delta x - x_{pb}},\ q_b = -\Delta y + m_b \cdot \Delta x$$

The equalisation of half-line f and half-line b delivers the ICR:

$$\begin{aligned} x_{ICR} &= \frac{q_b}{m_f - m_b} \\ y_{ICR} &= m_f \cdot \frac{q_b}{m_f - m_b} \end{aligned}$$

Finally, the distances f and b from the wheels to the ICR can be derived:

$$\begin{aligned} f &= \sqrt{x_{ICR}^2 + y_{ICR}^2} \\ b &= \sqrt{(x_{pr} - x_{ICR})^2 + (y_{pr} - y_{ICR})^2} \end{aligned} \tag{B.13}$$

In order to gain situation 4, only the running angle α has to be adjusted to

$$\alpha = \frac{v_b}{r} \cdot (t_4 - t)$$

Last, the phase times are determined in order to change the calculations for the speed ratios in time:

$$\begin{aligned} t_1 &= \frac{r \cdot \pi}{2 \cdot v_x} \\ t_2 &= t_1 + \frac{I}{v_x} \\ t_3 &= t_2 + t_{sym} \\ t_4 &= t_2 + 2 \cdot t_{sym} \\ t_5 &= t_4 + \frac{I}{v_x} \\ t_6 &= 2 \cdot t_3 \end{aligned} \tag{B.14}$$

Whereupon the symmetry time t_{sym} is derived as follows:

$$\Delta x_{sym} \overset{!}{=} \frac{a + I}{2} = I + s \cdot cos(\frac{\alpha_{sym}}{2})$$

and:

$$\alpha_{sym} = \frac{v_x}{r} \cdot t_{sym}$$

Thus:

$$t_{sym} = |\alpha_{sym} \cdot \frac{r}{v_x}| = |arcsin(\frac{I - \frac{a+I}{2}}{r}) \cdot \frac{r}{v_x}| \tag{B.15}$$

Appendix C

Feedforward and feedback control implementation

The implementation of the control concept is done through Python classes. This chapter provides an overview on the control classes and visualises their core function: the speed control. However, it cannot show the whole software structure since this would be too much code.

C.1 Feedforward code

In the following, the theoretical feedforward code examples for edges and I-profiles, discretisised by time t and wheel revolutions n, are presented, including a testfile which shows how the classes are called.
For real use, the control classes look slightly different in order to fit the software environment.

The Python class for concave edges, discretisised by time:

```
import math

class fefoconc:
    """FEEDFORWARD SPEED RELATIONS INSIDE CONCAVE EDGES"""


    ## CONSTRUCTOR
    def __init__(self, time_step, angle, obstacle_vel):

        self.dt      = time_step
        self.v       = obstacle_vel          # speed      [mm/s]
        self.old_v   = []          # list for the tracking of
            velocities

        # geometric properties
        self.a       = 50          # shaft centre distance      [
            mm]
        self.r       = 30          # wheel radius      [mm]
        self.g       = angle       # obstacle angle      [rad]

        # time state
        self.t = 0

        self.started = False


    ## START/STOP
    def start(self):
        self.started = True
        self.t = 0

    def stop(self):
        self.started = False


    ## VELOCITY COMMAND (AXIS CONTROL: FL[0]=FR[1] AND RR
        [2]=RL[3])
    def step(self, current_v):

        print self.t     # debug

        if self.started == False:
            return current_v

        if self.t < self.a/self.v:
            current_v[0] = self.v
            current_v[1] = current_v[0]
            current_v[2] = self.v*(-math.cos(self.g)+(self.v
                *self.t*(math.sin(self.g))**2)/math.sqrt(
                self.a**2-(self.v*self.t*math.sin(self.g))
                **2))
            current_v[3] = current_v[2]

        else:
            self.stop()

        self.t += self.dt          # timer

        self.old_v.append(current_v)          # add current_v
            to the list

        return current_v
```

The Python test class for concave edges:

```
import fefoconc as ffcc
import csv
import math
import time

time_step = 0.01
angle = .5*math.pi
obstacle_vel = 10
```

```
current_v = [10, 10, 10, 10]


robot = ffcc.fefoconc(time_step, angle, obstacle_vel)

robot.start()

while True:

    if robot.started == True:
        new_v = robot.step(current_v)

        with open('fefoconcLog.csv','a') as csvfile:
            fefoconcLog = csv.writer(csvfile)
            fefoconcLog.writerow([new_v[0],new_v[1],new_v
                [2],new_v[3]])

        current_v = new_v

    time.sleep(0.01)            # 10ms
```

The Python class for convex edges, discretisised by time:

```
import math

class fefoconv:
    """FEEDFORWARD SPEED RELATIONS INSIDE CONVEX EDGES"""


    ## CONSTRUCTOR
    def __init__(self, time_step, angle, obstacle_vel):

        self.dt         = time_step
        self.v       = obstacle_vel              # speed [mm/
            s]
        self.old_v      = []             # list for the
            tracking of velocities

        # geometric properties
        self.a          = 50             # shaft centre
            distance [mm]
        self.r          = 30             # wheel radius  [mm]

        # time state
        self.t = 0

        self.t1 = 0
        self.t2 = 0
        self.t3 = 0
        self.t4 = 0

        self.started = False
        self.set_angle(angle)
        self.update_phase_times()

    # Calculation of constant auxiliary variables s and
        a_sym
    def set_angle(self, new_angle):
        self.g       = new_angle
        self.s       = 2*self.r*(math.cos(self.g/2))**2
        self.a_sym   = ((-self.a*math.sqrt(2*(math.cos(self.g
            )+1)))/2+self.s)/math.sin(self.g)

    # Calculation of phase times
    def update_phase_times(self):
        self.t1 = self.r*(self.g-math.pi)/self.v
        self.t2 = self.t1 + self.a_sym/self.v
        self.t3 = self.t1 + 2*self.a_sym/self.v
        self.t4 = 2*self.t2


    ## START/STOP
    def start(self):
        self.started = True
        self.t = 0

    def stop(self):
        self.started = False


    ## VELOCITY COMMAND (AXIS CONTROL: FL[0]=FR[1] AND RR
        [2]=RL[3])
    def step(self, current_v):

        print self.t     # debug

        if self.started == False:
            return current_v

        # front wheels in curve between 1 and 2
        if self.t < self.t1:
            current_v[0] = self.v
            current_v[1] = current_v[0]
            current_v[2] = self.v*(math.cos(self.v*self.t/
                self.r)+math.sin(self.v*self.t/self.r)*(2*
                self.r*(math.sin(self.v*self.t/(2*self.r)))
                **2)/math.sqrt(self.a**2-(2*self.r*(math.sin
                (self.v*self.t/(2*self.r)))**2)**2))
            current_v[3] = current_v[2]

        # front and rear wheels outside 1 and 2
        elif self.t < self.t2:
            current_v[0] = self.v
            current_v[1] = current_v[0]
            current_v[2] = self.v*(-math.cos(self.g)-math.
                sin(self.g)*(-self.v*(self.t-self.t1)*math.
                sin(self.g)+(math.cos(self.g/2))**2*2*self.r
                )/math.sqrt(self.a**2-(-self.v*(self.t-self.
                t1)*math.sin(self.g)+(math.cos(self.g/2))
                **2*2*self.r)**2))
            current_v[3] = current_v[2]

        # half time! lead goes from front to rear

        # front and rear wheels outside 1 and 2
        elif self.t < self.t3:
            current_v[0] = self.v*(-math.cos(self.g)-math.
                sin(self.g)*(-self.v*(self.t3-self.t)*math.
                sin(self.g)+(math.cos(self.g/2))**2*2*self.r
                )/math.sqrt(self.a**2-(-self.v*(self.t3-self
                .t)*math.sin(self.g)+(math.cos(self.g/2))
                **2*2*self.r)**2))
            current_v[1] = current_v[0]
            current_v[2] = self.v
            current_v[3] = current_v[2]

        # rear wheels in curve between 1 and 2
        elif self.t < self.t4:
            current_v[0] = self.v*(math.cos(self.v*(self.t4-
                self.t)/self.r)+math.sin(self.v*(self.t4-
                self.t)/self.r)*(2*self.r*(math.sin(self.v*(
                self.t4-self.t)/(2*self.r)))**2)/math.sqrt(
                self.a**2-(2*self.r*(math.sin(self.v*(self.
                t4-self.t)/(2*self.r)))**2)**2))
            current_v[1] = current_v[0]
            current_v[2] = self.v
            current_v[3] = current_v[2]

        else:
            self.stop()

        self.t += self.dt                # timer

        self.old_v.append(current_v)             # add
            current_v to the list

        return current_v
```

The Python class for I-profiles, discretisised by wheel revolutions:

```
import math
# import numpy

class i_drive:
    """FEEDFORWARD SPEED RELATIONS FOR I-PROFILES,
        DISCRETISISED BY DRIVES"""
    ## v*t = n * 2r*pi -> n = v*t/(2r*pi)
    ## CONSTRUCTOR
    def __init__(self, obstacle_vel):

        self.v       = obstacle_vel        # speed      [mm/s]
        self.old_v   = []         # list for the tracking of
            velocities

        # geometric properties
        self.a       = 50.0         # shaft centre distance
             [mm]
        self.r       = 30.0         # wheel radius     [mm]
        self.I       = 10.0         # profile thickness [mm]
        self.g       = 1.5*math.pi       # obstacle angle     [
            rad]
        self.alpha   = 0

        # time state
        self.t = 0
        self.t1 = 0
        self.t2 = 0
        self.t3 = 0
        self.t4 = 0

        self.started = False
        self.update_phase_times()

    # Calculation of phase times
    def update_phase_times(self):
        self.n1 = (self.r/2*math.pi)/(2*self.r*math.pi)
        self.n2 = self.n1 + self.I/(2*self.r*math.pi)
        self.n3 = self.n2 + 2.1891833*self.v/(2*self.r*math.
            pi)
        self.n4 = self.n2 + 2*2.189183*self.v/(2*self.r*math
            .pi)
        self.n5 = self.n4 + self.I/(2*self.r*math.pi)
        self.n6 = 2*self.n3

        # print (self.n1, self.n2, self.n3, self.n4, self.n5
            , self.n6)

    ## START/STOP
    def start(self):
        self.started = True

    def stop(self):
        self.started = False


    ## VELOCITY COMMAND (AXIS CONTROL: FL[0]=FR[1] AND RR
        [2]=RL[3])
    def step(self, current_v, n):


        if self.started == False:
```

```
        return current_v

    # circ1 , front
    if n < self.n1:
        current_v[0] = self.v
        current_v[1] = current_v[0]
        current_v[2] = self.v*(math.cos(2*n*math.pi)+
            math.sin(2*n*math.pi)*(2*self.r*(math.sin(n*
            math.pi))**2)/math.sqrt(self.a**2-(2*self.r
            *(math.sin(n*math.pi))**2)**2))
        current_v[3] = current_v[2]

    # straight , front
    elif n < self.n2:
        current_v[0] = self.v
        current_v[1] = current_v[0]
        current_v[2] = self.v*(-math.cos(self.g)-math.
            sin(self.g)*(-2*self.r*math.pi*(n-self.n1)*
            math.sin(self.g)+(math.cos(self.g/2))**2*2*
            self.r)/math.sqrt(self.a**2-(-2*self.r*math.
            pi*(n-self.n1)*math.sin(self.g)+(math.cos(
            self.g/2))**2*2*self.r)**2))
        current_v[3] = current_v[2]



    ### I, front ####
    elif n < self.n3:
        self.alpha = 2*self.r*math.pi*(n-self.n2)/self.r
                # numpy.float64()
        self.s = 2*self.r*math.sin(self.alpha/2)

        # frot wheel - point on curve
        self.deltax = self.I + self.s*math.cos(self.
            alpha/2)
        self.deltay = self.r - self.s*math.sin(self.
            alpha/2)

        # rear wheel - point on curve
        self.x = -(self.deltay*math.sqrt(-self.r**4+2*(
            self.a**2+self.deltax**2+self.deltay**2)*
            self.r**2-self.a**4+2*self.a**2*(self.deltax
            **2+self.deltay**2)-self.deltax**4-2*self.
            deltax**2*self.deltay**2-self.deltay**4)-
            self.deltax*(self.r**2-self.a**2-self.deltax
            **2-self.deltay**2))/(2*(self.deltax**2+self
            .deltay**2))
        self.y =  (self.deltax*math.sqrt(-self.r**4+2*(
            self.a**2+self.deltax**2+self.deltay**2)*
            self.r**2-self.a**4+2*self.a**2*(self.deltax
            **2+self.deltay**2)-self.deltax**4-2*self.
            deltax**2*self.deltay**2-self.deltay**4)+
            self.deltay*(self.r**2-self.a**2-self.deltax
            **2-self.deltay**2))/(2*(self.deltax**2+self
            .deltay**2))

        # intersection of f and b
        self.mf = 1/math.tan(self.alpha)
        self.mb = (-self.deltay-self.y)/(-self.deltax-
            self.x)
        self.qb = -self.deltay + (self.mb*self.deltax)

        self.xs = self.qb/(self.mf-self.mb)
        self.ys = self.mf*self.qb/(self.mf-self.mb)

        # distances to ICR - f and b
        self.f = math.sqrt(self.xs**2+self.ys**2)
        self.b = math.sqrt((self.x-self.xs)**2+(self.y-
            self.ys)**2)

        # velocities
        current_v[0] = self.v
        current_v[1] = current_v[0]
        current_v[2] = self.v*self.b/self.f
        current_v[3] = current_v[2]

# half time! lead goes from front to rear

### I, rear ###
elif n < self.n4:

        self.alpha = 2*self.r*math.pi*(self.n4-n)/self.r
        self.s = 2*self.r*math.sin(self.alpha/2)

        # frot wheel - point on curve
        self.deltax = self.I + self.s*math.cos(self.
            alpha/2)
        self.deltay = self.r - self.s*math.sin(self.
            alpha/2)

        # rear wheel - point on curve
        self.x = -(self.deltay*math.sqrt(-self.r**4+2*(
            self.a**2+self.deltax**2+self.deltay**2)*
            self.r**2-self.a**4+2*self.a**2*(self.deltax
            **2+self.deltay**2)-self.deltax**4-2*self.
            deltax**2*self.deltay**2-self.deltay**4)-
            self.deltax*(self.r**2-self.a**2-self.deltax
            **2-self.deltay**2))/(2*(self.deltax**2+self
            .deltay**2))
        self.y =  (self.deltax*math.sqrt(-self.r**4+2*(
            self.a**2+self.deltax**2+self.deltay**2)*
            self.r**2-self.a**4+2*self.a**2*(self.deltax
            **2+self.deltay**2)-self.deltax**4-2*self.
            deltax**2*self.deltay**2-self.deltay**4)+
            self.deltay*(self.r**2-self.a**2-self.deltax
            **2-self.deltay**2))/(2*(self.deltax**2+self
            .deltay**2))
        # intersection of f and b
        self.mf = 1/math.tan(self.alpha)
        self.mb = (-self.deltay-self.y)/(-self.deltax-
            self.x)
        self.qb = -self.deltay + (self.mb*self.deltax)

        self.xs = self.qb/(self.mf-self.mb)
        self.ys = self.mf*self.qb/(self.mf-self.mb)

        # distances to ICR - f and b
        self.f = math.sqrt(self.xs**2+self.ys**2)
        self.b = math.sqrt((self.x-self.xs)**2+(self.y-
            self.ys)**2)

        # velocities
        current_v[0] = self.v*self.b/self.f
        current_v[1] = current_v[0]
        current_v[2] = self.v
        current_v[3] = current_v[2]



    # straight rear
    elif n < self.n5:
        current_v[0] = self.v*(-math.cos(self.g)-math.
            sin(self.g)*(-2*self.r*math.pi*(self.n5-n)*
            math.sin(self.g)+(math.cos(self.g/2))**2*2*
            self.r)/math.sqrt(self.a**2-(-2*self.r*math.
            pi*(self.n5-n)*math.sin(self.g)+(math.cos(
            self.g/2))**2*2*self.r)**2))
        current_v[1] = current_v[0]
        current_v[2] = self.v
        current_v[3] = current_v[2]

    # circ2, rear
    elif n < self.n6:
        current_v[0] = self.v*(math.cos(2*math.pi*(self.
            n6-n))+math.sin(2*math.pi*(self.n6-n))*(2*
            self.r*(math.sin(math.pi*(self.n6-n)))**2)/
            math.sqrt(self.a**2-(2*self.r*(math.sin(math
            .pi*(self.n6-n)))**2)**2))
        current_v[1] = current_v[0]
        current_v[2] = self.v
        current_v[3] = current_v[2]

    else:
            self.stop()
            print "fertig"


    self.old_v.append(current_v)        # add current_v
        to the list

    return current_v
```

C.2 Feedback code

The Python class for feedback speed control:

```
"""
What is a reasonable margin for the torque difference
    between front and rear?
What is a reasonable change in velocity "deltavel"?
What is a reasonable sleep time?
-> EXPERIMENTS!
"""

import math
from control_blocks import ControlBlock

class feba(ControlBlock):
    """FEEDBACK SPEED RELATIONS INSIDE OBSTACLES"""

    def configure(self):
        self.limit           = self.options['margin']
        self.dv              = self.options['delta_vel']


    ## TORQUE CONTROL (AXIS CONTROL: FL[0]=FR[1] AND RR[2]=
        RL[3])
    def step(self, **kwargs):

        current_v        = kwargs['planned_speeds']
        current_torque   = kwargs['current_torques']

        self.dtorque = (-current_torque[0]+current_torque
            [1]) - (current_torque[2]-current_torque[3])
        #print self.dtorque #          debug

        # rear wheels "too slow"
        if  self.dtorque > self.limit:
            print "rear wheels too slow:"
            print current_torque
            current_v[2] = current_v[2] + self.dv*self.
                dtorque
            current_v[3] = current_v[3] + self.dv*self.
                dtorque

        # rear wheels "too fast"
        elif -self.dtorque > self.limit:
            print "rear wheels too fast:"
            print current_torque
            current_v[2] = current_v[2] + self.dv*self.
                dtorque
            current_v[3] = current_v[3] + self.dv*self.
                dtorque
```

Appendix D

Testing results

This chapter addresses selected items of the test plans of Chapter 3.3.1. The numbers are directly referred to Table 3.1 and Table 3.2 respectively.
The following torque plots show the motor end torques for the four motors front left (FL), front right (FR), rear right (RR) and rear left (RL).

In addition, an experimental analysis of the feedforward control failure for concave edges of $\gamma = \frac{\pi}{2}$ is done at the end of the chapter. As a side effect, it provides a couple of deeper insights into the robot behaviour inside obstacles.

D.1 Torque output (No.2) and Torques and gravity (No.3)

Torques are almost equal for all wheels while driving horizontally (see Figure D.1). The small aberrations are most likely due to imprecisions of the wheel alignment.

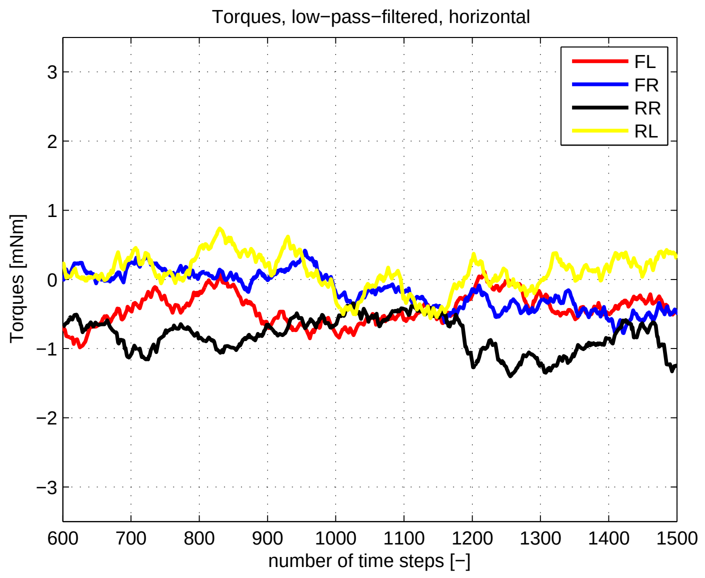


Figure D.1: Horizontal torques.

Gravity influences the torques as it was to be expected: Torques increase while driving upwards and they decrease while driving downwards (cf. Figure D.2).

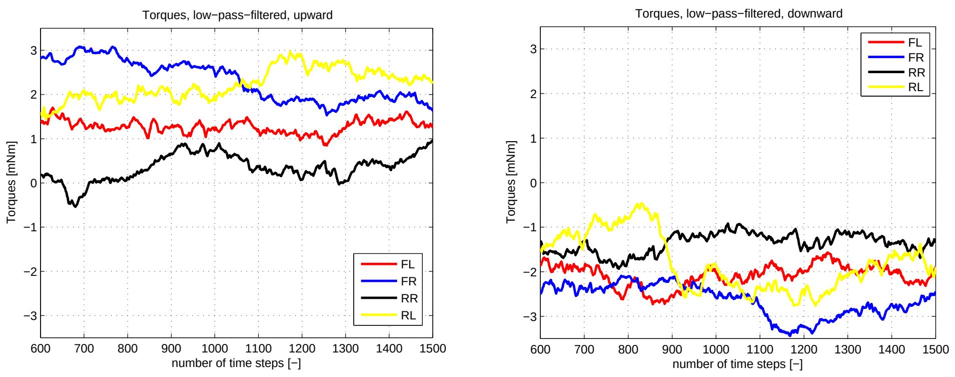


Figure D.2: Upward and downward torques.

D.2 Torques and curves (No.4)

The influence of curves on the torque behaviour is significant and quantitatively comparable to the obstacle influence (see Figure D.3). The torque increase is mainly due to the shear forces between the wheels and the surface. They cannot be countered by adjusting wheel speeds. Therefore, Obstacle Climbing Control for curves is not suitable.

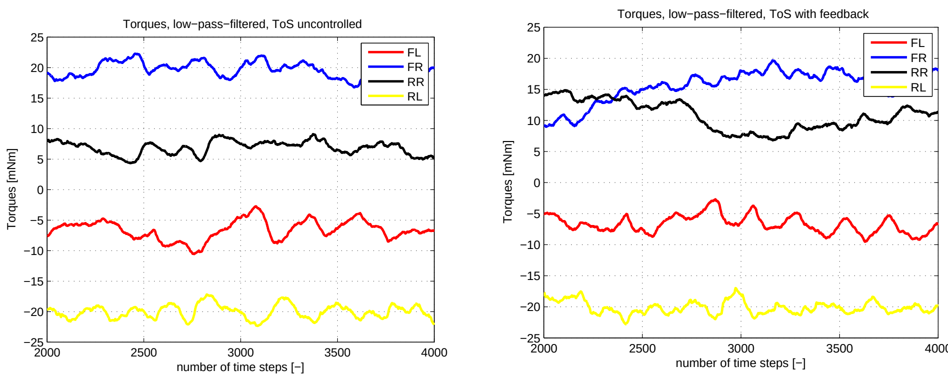


Figure D.3: Uncontrolled and feedback turn on spot.

D.3 Torques and obstacles (No.5) and Structural stress (No.10)

Out of several test runs, the one in Figure D.4 is chosen to illustrate the testing results for the obstacle influence on structural stress. Obstacles 1 (around 500 time steps) and 3 (around 2000 time steps) are concave edges of $\gamma = \frac{3}{4}\pi$. Number 2 (around 1250 time steps) is a convex edge of $\gamma = \frac{3}{2}\pi$.

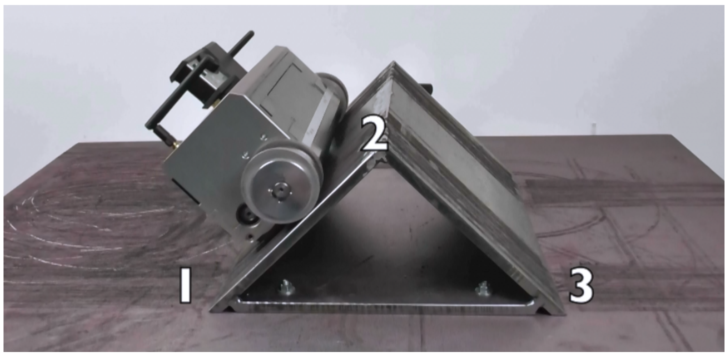


Figure D.4: Reference test run.

D.3.1 Uncontrolled torque behaviour

The uncontrolled torque plot proves that obstacle climbing causes high torque peaks (Figure D.5).

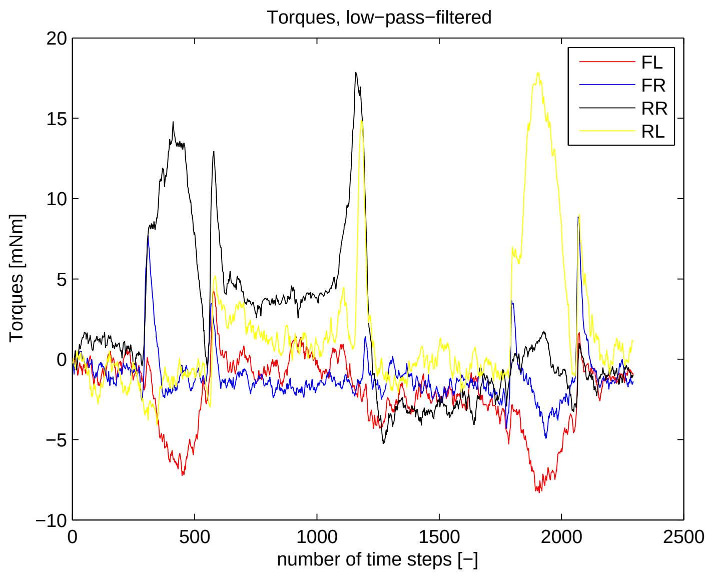


Figure D.5: Uncontrolled test run.

D.3.2 Feedforward torque behaviour

For the sake of an accurate feedforward control algorithm activation, only the convex edge (Figure D.6, around 1250 time steps) is controlled in the following plot. Further experiments prove the same torque minimising effect for concave edges.

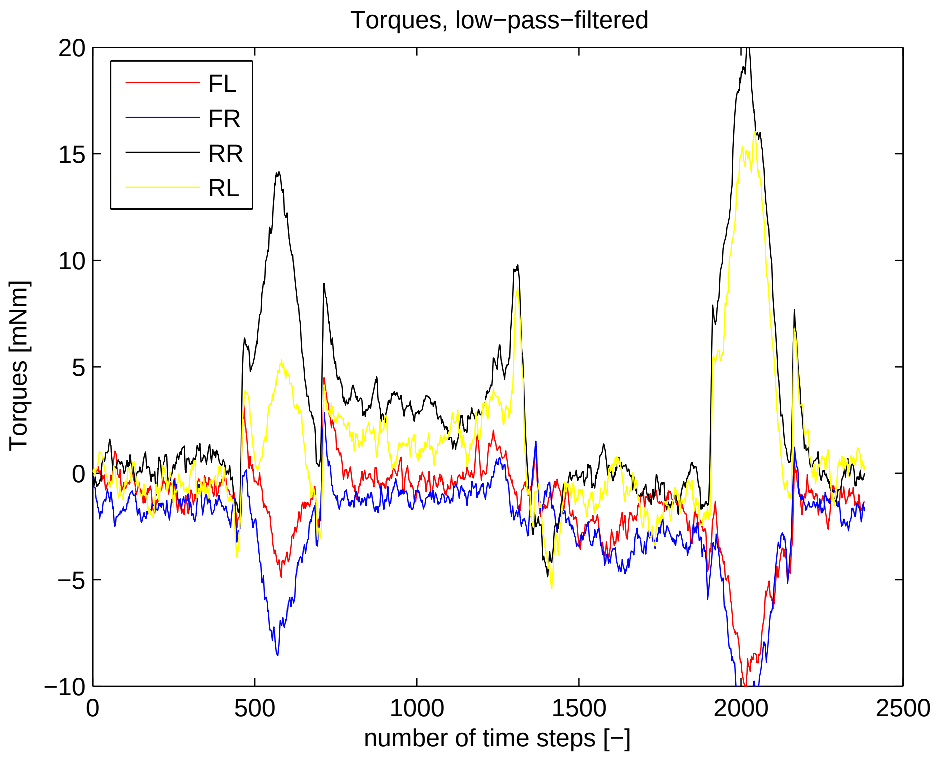


Figure D.6: Feedforward test run.

D.3.3 Feedback torque behaviour

For previously discussed reasons, feedback control is not very effective inside convex edges with γ close to $\frac{3}{2}\pi$ (cf. Chapter 4.4.1). The nice impact inside concave edges is shown in Figure D.7.

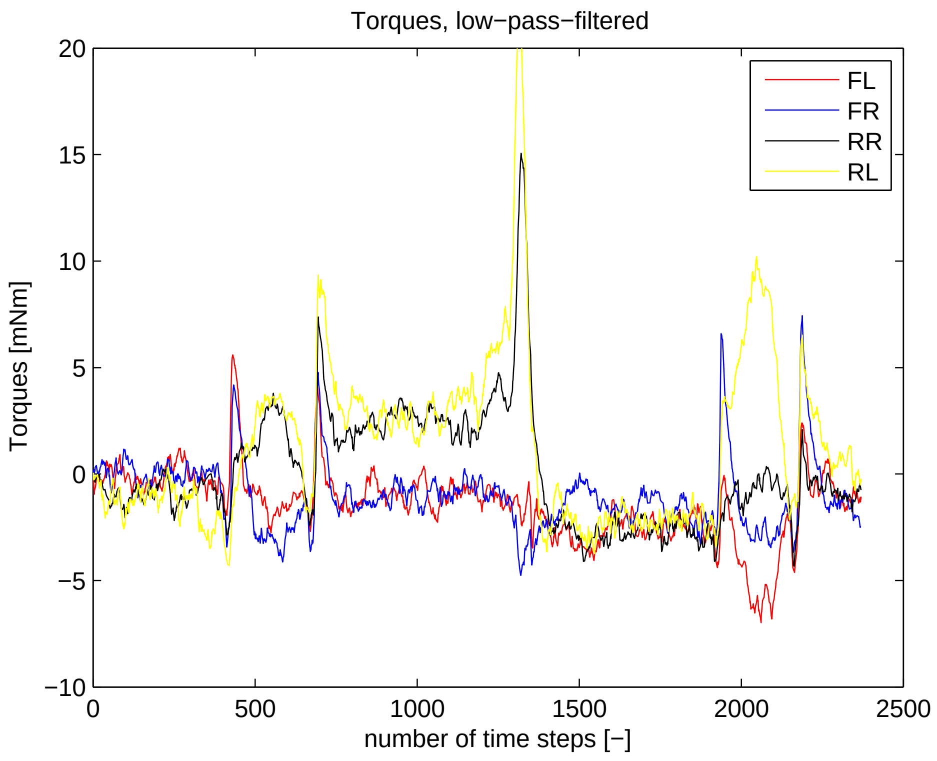


Figure D.7: Feedback test run.

D.4 Speed accuracy (No.7)

With feedforward control, speeds agree well in theory and during test runs. The only problems occur when speeds change rapidly and abruptly (cf. Figure D.8 (right), around 1380 time steps). The use of motors with higher accelerations would help to solve this problem.

The difficulties inside concave edges are addressed in Appendix D.6.

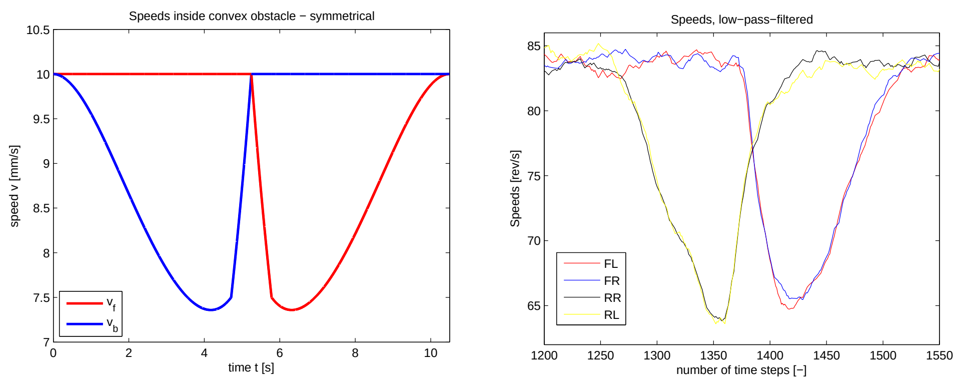


Figure D.8: Feedforward speed comparison for $\gamma = \frac{3}{2}\pi$.

On the contrary, the feedback controlled speeds inside convex edges only fit theory in shape (cf. Figure D.9). This does not serve for more than a rudimentary confirmation of the second part of the feedback hypothesis (cf. Chapter 4.4). The speed control based on the torque comparison is too slow and too weak respectively. As a consequence, the impact of feedback control inside convex edges is not useful enough.

By the way, the left hand plot in Figure D.9 is the same one than the left hand plot of Figure D.8. The only difference is that symmetry is not used. The lead stays on the front wheels permanently. Since feedback control always adjusts the rear wheel speeds while leaving the front wheels untouched at constant speed, the according speed plots cannot be symmetrical but agree with this newly introduced plot.

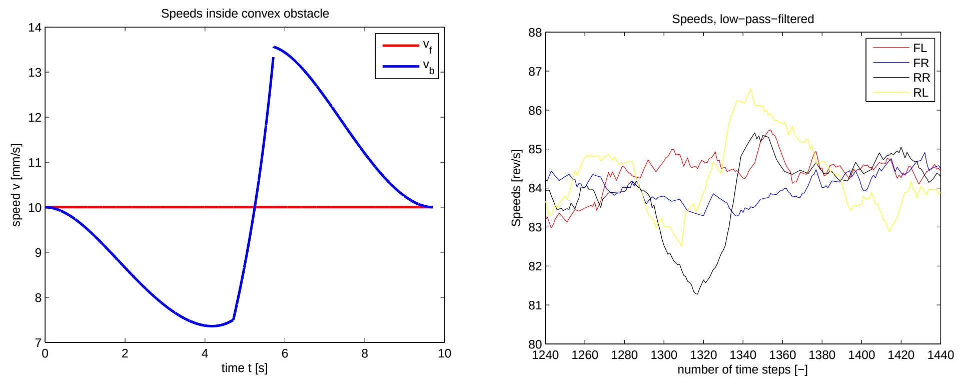


Figure D.9: Feedback speed comparison for $\gamma = \frac{3}{2}\pi$.

D.5 Wheel slip (No.9)

Torque peaks and wheel slip cannot occur at the same time. For the test runs of Chapter D.3 it is to prove that wheel slip is negligible. More importantly, wheel slip should be the same for uncontrolled and controlled test runs because it is not the aim of the control approaches to reduce the torques at the expense of wheel slip.
Table D.1 shows the values of the slip test runs (cf. Figure D.4). The measured distance is the horizontal difference between the starting and the end point of the robot. It is captured by a leaser measurement device. Including the obstacle geometry delivers the real distance which is covered by the robot (see Figure D.10):

$$d_{real} \;=\; d_{measured} + (2 \cdot 210 - 300 - 4 \cdot r \cdot tan\frac{\pi/4}{2} + \frac{2r\pi}{4})\,mm = d_{measured} + 117\,mm$$

The motor revolutions are the difference between the number of encoder counts at the starting and the end point. The robot output distance is calculated by multiplying the motor revolutions with the gear ratio and the wheel circumference:

$$d_{out} = rev_{mot} \cdot \frac{1}{198 \cdot \frac{16}{12}} \cdot 2r\pi$$

Test run	**Measured distance [mm]**	**Real distance [mm]**	**Motor revolutions [rev]**	**Output distance [mm]**
Uncontrolled				
1	665	782	1098	784
2	660	777	1088	777
3	658	775	1086	775
Average	661	778	1091	779
Wheel slip: **0.13 %**				
Feedforward				
1	653	770	1080	771
2	682	799	1118	798
3	663	780	1092	780
Average	666	783	1097	783
Wheel slip: **0.00 %**				
Feedback				
1	653	770	1082	773
2	654	771	1083	773
3	658	775	1087	776
Average	655	772	1084	774
Wheel slip: **0.26 %**				

Table D.1: Slip measurements.

A comparison of the effective measured distance and the output distance leads to the wheel slip: $slip = \frac{d_{out} - d_{real}}{d_{real}}$. The test runs confirm that wheel slip is negligible. As the values do not deviate significantly, three test runs per control approach are considered to be enough.

A film of the slip test run reveals which kind of obstacles cause slip. At the end of concave obstacles the front wheels slip slightly. More precisely, the slip occurs when the rear wheels are exactly inside the edge and do not want to detach. Therefore, they hold the front wheels back for a short moment and make them slip before the motion goes on. No visible slip is observed inside convex edges.

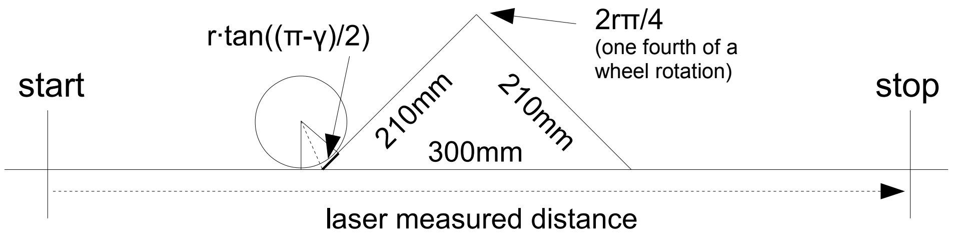


Figure D.10: Geometry of the slip test track.

D.6 Deeper insights into the robot behaviour inside obstacles

The feedforward control approach for concave edges of $\gamma = \frac{\pi}{2}$ fails because the theoretical speeds become infinite (see Figure 3.3 (left)). Figure D.11 shows the torque (left) and speed (right) behaviour for an uncontrolled test run. The plots confirm theoretical considerations. In a first step, the rear wheels have to cover less distance than the front wheels (cf. Figure 1.1). Therefore, they are forced to slow down. The rear wheel torques increase significantly because the motor controllers try to keep the higher set speeds and thus the rear wheels push the front wheels. Consequently, the front wheel torques decrease because part of their load is taken by the rear wheels. In a second step after "halftime" (around 930 time steps), the contrary situation arises.

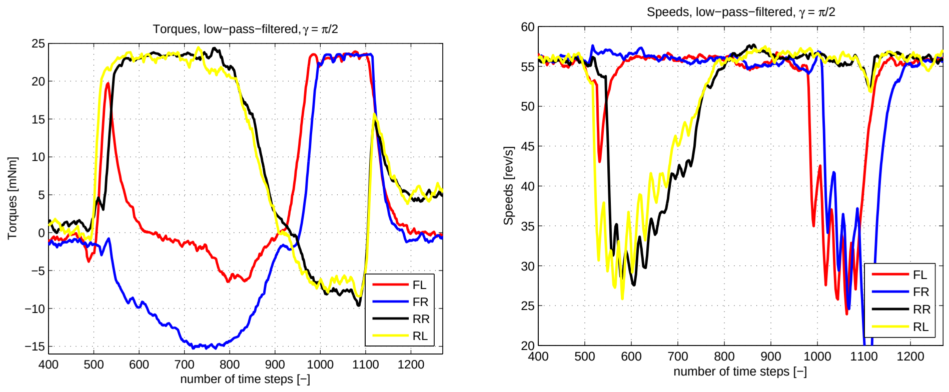


Figure D.11: Robot behaviour inside concave edges of $\gamma = \frac{\pi}{2}$.

This behaviour reveals and confirms two key findings of this thesis. Firstly, the torque behaviour prediction of Chapter 4.1.5 is true. Feedback control based on a torque comparison is possible.

Secondly, speeds inside obstacles behave like in theory and they behave symmetrically. Therefore, a successful control approach should use symmetry as well. The feedback performance could probably be enhanced with the enabling to switch the leading wheels. This would automatically end in a symmetrical behaviour.

The symmetrical feedforward solution for concave edges of $\gamma = \frac{\pi}{2}$ (see Figure D.12 (left)) agrees with the uncontrolled speed behaviour (cf. Figure D.11 (right)). This analogy delivers further reasons for the torque behaviour inside obstacles and additional explanations how the feedforward approach works.
Feedforward control causes the same speed ratios like without control. However, these speed ratios are no longer forced by the obstacle geometry but set with feedforward control. Hence, the motor controllers do not fight against these set speeds. Unnecessarily high torques can be prevented.
In the case of the compact and stiff Ship Inspection Robot, the obstacles automatically force the correct speed ratios and thus cause torque peaks. For a more flexible designed robot, neither torque peaks nor the correct speeds would be forced but the robot would just become distorted.

Last, the connection between concave and convex edges is established in more detail. A comparison of Figure D.12 (left) and Figure 3.3 (right) reveals how the obstacles are linked: The speed ratios within the middle phases 2 and 3 of the convex edge agree with the concave edge speed ratios.
Within the phases 1 and 4 inside convex edges the front wheels and the rear wheels respectively are on a circular path (red) and follow their extra speed ratios. Once all the wheels are outside the starting and the end point of the circular path, the convex and the concave situation are the same (cf. Figure D.12 (right)).

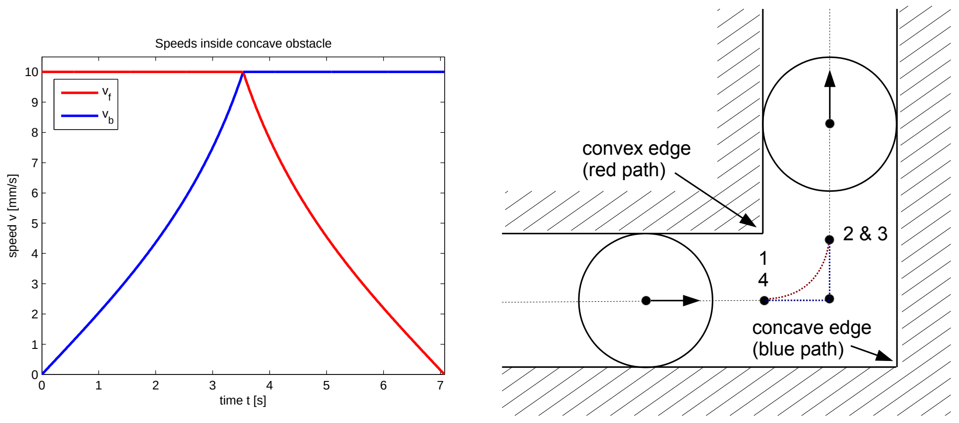


Figure D.12: Symmetrical speed ratios for concave edges of $\gamma = \frac{\pi}{2}$ and obstacle comparison.

Bibliography

[1] Florian Berlinger et al. *Ship Inspection Robot.* Tech. rep. ETH Zurich, 2013.

[2] Wolfgang Fischer. "Design of compact climbing robots for power plant inspection". PhD thesis. ETH Zurich, 2010.

List of Tables

List of Figures