Department of Electronic Systems

Titel:
Modelling and Control of a six legged mobile robot
Theme of the Semester:
Intelligente autonome systemer
Project Periode:
Master Thesis, 1. September 2009 - 23 of June 2010
Project Group:
1039

Authors:
Sigurd Villumsen

Advisor:
Mads Sølver Svendsen

Number of copies: 5
Number of pages:
In Main Report: 168
In Appendix Report: 38
Total Number of pages: 216
Submission date: 23 of June 2010

Abstract:

The following master thesis concerns the modelling and control of an autonomous, six legged mobile robot, with 18 degrees of freedom. Three models were made, the first being a kinematic model, describing the position of the limbs of the robot based on the angles of the servo motors. The second being an inverse kinematic model, which derived the angles of the servo motors to produce a given end point position of the legs. And the third being a dynamic model, based on the iterative Newton-Euler approach, combined with the dynamics of a rigid free body. This dynamic model was however not verified due to the lack of sensors on the robot. A software system was designed to make the mobile robot (MR) able to take decisions regarding path planning, and a computer vision system was implemented to make the MR capable of line tracking. Finally a gait generator was constructed, and based on this, a controller was implemented. The controller was able to make the MR track the black line, but the designed software system was not implemented due to hardware issues.

Afdeling for Kontrol

Titel:
Modellering og kontrol af en seksbenet mobil robot
Semester Thema:
Intelligente autonome systemer
Projektperiode:
Master Thesis, 1. September 2009 - 23. juni 2010
Projektgruppe:
1039

Forfattere:
Sigurd Villumsen
Vejleder:
Mads Sølver Svendsen

Antal kopier: 5 stk
Antal sider:
I hoved rapporten: 168
I appendiks rapporten: 38
Antal sider ialt: 216
Afleveringsdato: 23. juni 2010

Synopsis:

Det følgende speciale omhandler modelleringen og reguleringen af en autonom seksbenet mobil robot med 18 frihedsgrader. Tre modeller blev udarbejdet. Den første, som er den kinematiske model, beskriver positionen af robottens led ud fra vinklerne på robottens servomotorer. Den anden er en invers kinematisk model, som beskriver, hvilke vinkler, der er nødvendige for at opnå en given position af robottens ben. Den tredje model er en dynamisk model baseret på en iterativ Newton-Euler fremgangsmåde koblet sammen med en "free body". Denne sidste model blev dog ikke verificeret grundet manglende sensorer. Et software system blev designet for at gøre robotten i stand til at tage beslutninger omhandlende ruteplanlægning, mens et computer vision system blev implementeret for at gøre robotten i stand til at følge en linje. Endelig blev en "gait" generator konstrueret, og ud fra denne blev en regulator fremstillet. Regulatoren gjorde robotten i stand til at følge en sort linje, men det designede software system blev ikke implementeret grundet hardware problemer.

Preface

This master thesis has been produced by Sigurd Villumsen and describes the obtained results of the final project at Aalborg University, within the specialization of Intelligent Autonomous Systems. The project has been partly written in Denmark at Aalborg University, under the department of Electronic Systems, and partly in Croatia, at University of Zagreb under FER (Fakultet elektrotehnike i racunarstva - Faculty of Electrical Engineering and Computing). The thesis has been written under the supervision of Mads Sølver Svendsen. Its workload is nominated to 48 ECTS, and it has been completed during a time period of two semesters.

Reading Guide

It has been chosen to divide the main report into five parts. Part I contains an introduction and an analysis of the mobile robotic system, followed by a description of the obstacle course on which it should operate. This description leads to a requirements specification, and based on these requirements the mobile robot is constructed. Part II of the report contains the design and verification of a mathematical model, and a description of stability. In part III the general software design is described, a computer vision system is developed, and a path finding algorithm is implemented. Part IV describes the generation of an optimal gait, on a basis of a developed performance function. In the final part V the designed system is tested, and a conclusion is made.

References to secondary literature will be done by [Page,nr], or simply [nr]. This refers to a full list of enumerated sources located in the back of the report.

The Author of the Thesis

Sigurd Villumsen

Contents

Part I

Preliminary analysis

Chapter 1

Introduction

In the following sections, an introduction to the master thesis will be given. The purpose of the project will be described followed by a requirements specification which leads to the selection of the mobile robot.

1.1 Defining project purpose

The following thesis is inspired by the massive amount of attention the concept of legged robotics has obtained until now.

1.1.1 Advantages of legged locomotion

One of the main reasons for the great interest in legged locomotion is because it offers some very profound advantages over wheeled locomotion, some of which are described in [1, p17-19], [2, p28-32] and [3]. According to [3], most of the earths land surface is inaccessible to vehicles using regular propulsion (wheels). This is due to the fact that wheeled locomotion requires ground contact support along the entire path of motion. In rough terrains, physical support cannot be guaranteed in the entire path of motion due to the existence of various obstacles. This can to some extent be improved by for example using treaded vehicles, but still a continuous path is needed. For legged locomotion, only a set of footholds is required, and not a continuous path. The quality of the ground between these footholds is unimportant provided the MR has enough ground clearance. This entails that when traversing rough terrain, legged locomotion is superior to wheeled locomotion when considering accessibility. In [1, p15], the following figure can be found From this figure it is seen that wheeled

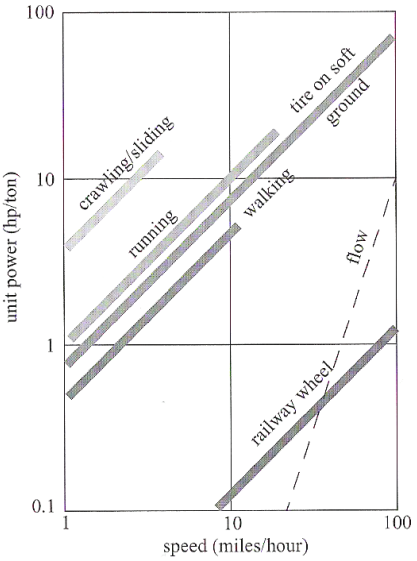

Figure 1.1: Shows the power consumption of various means of locomotion, as a function of speed

locomotion on a flat, frictionless surface is far more effective power consumption wise than legged locomotion (on the figure depicted as a railway wheel). It does however also show that when a tire operates on soft ground, legged locomotion is suddenly also more energy efficient. This fact can also be seen in nature where many animals and insects use legs as a form of locomotion. There exist various types of legged locomotion, among others configurations with one, two, four, six and more legs. Examples of such robots can be seen from figures 1.2 to 1.5[4],[5],[6],[7].

1.1.2 Hexapod robot

A hexapod robot is a robot which achieves locomotion by means of a configuration of six legs, as seen from figure 1.5. Hexapod robots are very useful for developing mobile robotic platforms, as they, with six legs can obtain a great deal of stability when walking. According to [8, p1] they are also favoured as they can provide a good platform, capable of carrying more weight than eg. a biped robot. Many articles have also mentioned, that due to the many benefits of hexapod robots, they could prove very useful when operating in various disaster areas, such as collapsed building. [9],[10] and [11].

Many of these publications are however only focused on one thing - the gait of the system. Instead, this thesis will be focused on developing an autonomous system as a whole, based on a hexapod robot, and it will cover many of the fundamental problems, when dealing with autonomous systems.

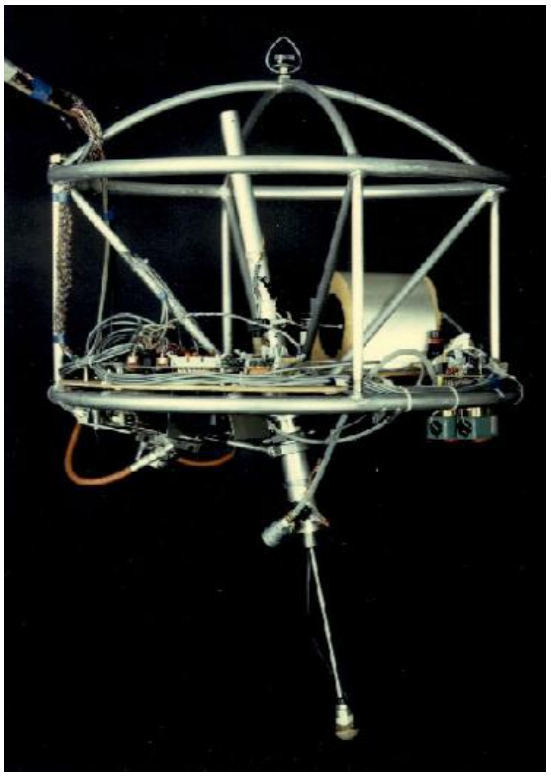

Figure 1.2: Shows a one legged jumping robot, called the Raibert Hopper

Figure 1.3: Shows a two legged robot developed by Honda, called ASIMO

Figure 1.4: Shows the four legged robot big dog developed by boston dynamics

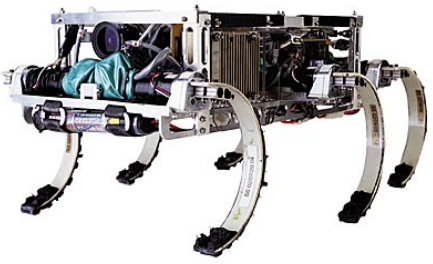

Figure 1.5: Shows the six legged robot called Rhex also developed by boston dynamics

Every year in April, a competition for autonomous mobile robots called RoboCup is held at the Technical University of Denmark (DTU), where the author of this report has been a participant for the last three years (2010, 2009 and 2008) with Team Rubberduck. The general concept of the competition is, that an autonomous mobile robot (MR) moves along a black line, while solving various assignments to obtain points (a more thorough description will be given in section 2). Each year new tasks are added to increase the difficulty of the

track, but also to make the competitors come up with new clever solutions to general problems in the design of autonomous systems. This indicates that the RoboCup track provides a good test of the capabilities of autonomous mobile robots. To have a more concrete task for the autonomous MR in this study, it is designed to operate on the track of the RoboCup.

Now that the general purpose of the thesis has been outlined, it is necessary to get a clear understanding of the task at hand. To provide this understanding, the following chapter will be devoted to describing the track of the RoboCup competition.

Chapter 2

Robocup

As described in the previous section, an autonomous mobile robot should be constructed to operate in an environment similar to the track of the RoboCup competition. To give some a priori information about the competition, a small description of the RoboCup competition is made in the following chapter, describing the concepts of the competition, what manner of assignments the obstacle course provides, and the point system.

2.1 General concepts

Generally the RoboCup competition is a competition for mobile robots (MR), with the ability to track lines. The MR needs to follow a black line on an obstacle course to come from the start position to the end position. While following this path, the robot passes under several yellow gates, and can accomplish several smaller missions. Solving the missions or moving under the yellow gates, provides the MR with points. The MR with the highest amount of points wins the competition.

2.2 Obstacle course

The obstacle course of the RoboCup competition can be seen from section 2.1 From this figure, one can see that the obstacle course is literally divided in two, that is, to pass under all yellow gates, and accomplish all missions, the MR needs to be able to move away from the black line. Another thing that is noted is that the surface, on which the black line is placed, consists of two colours, grey and white. This entails that the MR should be robust enough so that navigation on both the grey and white areas are feasible. The numbers on Figure 2.1 are

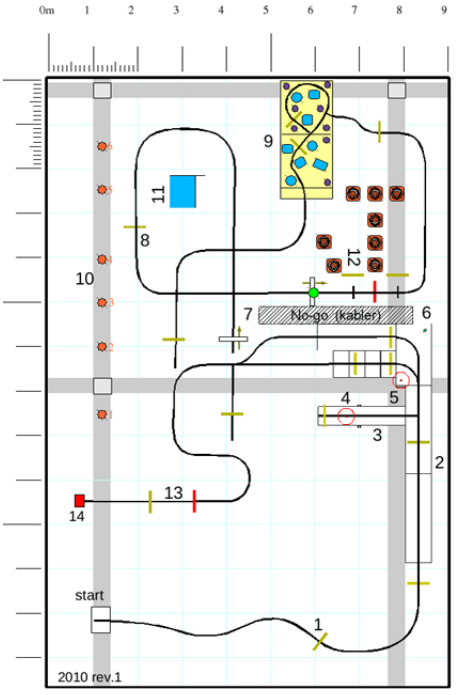

Figure 2.1: Shows the obstacle course on which the MR needs to move.

numbers indicating the various missions, which the MR can undertake. Generally the MR should move from start to goal (14), while collecting as many points as possible. The possible points and their associated missions are described below, with references to the numbers they are represented by. These descriptions are found from [12]. The points obtained from ports are generally not described below, but can be seen from the figure instead

- **Black line**
  The black line is made of black dull tape (type Tesa-4651), and is placed approximately as shown on figure 2.1, in such a way, that no curve will be sharper than a circle with a radius of 50 cm. The black tape is visible both on the light floor and the tiles.

- **Gates(± 1 point each)**
  The gates on the obstacle course can either be red or yellow. yellow gates gives one point, while red ones give - one point. Both of them are 45cm wide($\pm$2cm) and min 47 cm high.

- **Start**
  The start position is on top of a pressure plate(bottom left corner of the figure). When the MR moves from the plate, a timer starts, and after a certain amount of time, a port(1) closes. To pass through the gate, an average speed of 25 cm/s is needed.

- **Ramp and seesaw**
  The ramp leads the MR to a plateau which is 55 cm high. The inclines over a distance

of 50 cm, which gives an incline of approximately 9 degrees. A seesaw (3) is mounted on the left side of the ramp's highest part. The seesaw's normal position is horizontal, and it is supported at the end of the ramp. This means that the seesaw will remain horizontal when a vehicle is driving from the ramp down to the seesaw until the vehicle is passing the supporting point. The transition from the ramp to the seesaw will have a height difference of approximately 6 cm.

- **Golf balls(1 point each)**
  At the positions (4) 30 cm from the seesaw's supporting point and (5) on the ramp's plateau, two red golfballs are placed. If the MR pics one up, and puts it in the hole (6) one point is gained.

- **Ramp/steps down**
  There exists two ways down from the plateau, one down a ramp, with a descent of approximately -15 degrees, or down 4 steps 40 cm long, with an approxamate descent of 11 cm each. On the ramp, the MR passes one and on the stair two yellow gates.

- **Racetrack(3 points)**
  The racetrack (7) starts and ends with an electronic gate. When a beam of light is broken in the start gate, the time measurement is started, and it stops when a corresponding beam is broken in the end gate. A yellow gate (8) is placed on the line between the start and end gates. This yellow gate has to be passed after the start gate and before the end gate to obtain points on the racetrack. The yellow gate gives one point, and depending on the speed further zero, one, two or three points can be achieved. The time limits are not fixed, but it will require an average speed of more than 1m/s to obtain maximum points at this challenge.

- **Hump track**
  The hump track (9) is an area of 120 by 240 cm with humps and holes. The hump track is raised approx 2 cm above the floor level. The approach ramp has approx the same gradient as the ramp at (2). The holes are approx 12 cm in diameter and approx 2 cm deep (the thickness of the plate). The size of the humps will vary from 0 at the edge to max approx 8 cm.

- **Flowers($\pm$1 point)**
  At (10) there is 6 positions marked with stars, on two of these positions there will be placed a (plastic) pot plant. The flowers will be placed at random on two of these positions after start. One of the flowers is a red begonia in a reddish pot pot and the other a blueish veronica in a black pot. If the blueish flower is brought to the goal, one point is obtained, if the reddish one is brought, one point is deducted.

- **Tunnel (2 points)** The tunnel (11) is mounted with an outward opening door at each end, each 25 cm high and 45 cm wide. The front door is broader than the frame - so that it can be pushed open from the outside. You obtain one point for passing each of the doors.

- **Bowling alley**
  The bowling alley (12) consists of nine skittles, and is placed abreast of a red gate (giving 1 penalty point) and in between two side lines placed 1 m apart. Two yellow gates are placed opposite the two side lines 40 cm from the main line. To obtain the maximum amount of points, the MR needs to drive through the first yellow gate, and around the 9 skittles, and back on the black line through the last gate. An area (13) marked with white tape to the left of the red gate is forbidden and must thus not be used by the robot. If just one wheel of the robot touches this area, you obtain one penalty point.

- **Blinking gates**
  The blinking gates(13) consist of ordinary yellow gates, and on each of them a light indication is mounted consisting of four read stoplight arrays. When starting, the lights on one(chosen randomly each run) of the gates, starts blinking the light indication will flash approx once a second (switched on 0.5s, switched off 0.5s). If the MR passes the flashing gate one point is deducted, while passing the other gate (with the light indication switched off) results in one recieved point.

- **Goal siren**
  The goal siren (15) is activated by pressing the front plate (which activates a switch mechanically). The width of the front plate is approx 15 cm, the height 10 cm, and it is lowered approx 1 cm in proportion to the frame. Reaching the goal (from the front side), you obtain two points.

- **Sequence**
  Apart from the guillotine (1) and the goal (14), the sequence of the obstacles may be chosen arbitrarily and it is not required that all obstacles have been passed.

This entails that generally, the types of assignments can be divided into two categories.

- Node missions
- Path missions

In the first type of missions the MR needs to get to a certain node, and do some sort of action eg. moving to the golf ball, and picking it up. In the second type of missions, the path

missions the robot needs to travel along a given path. like moving down the stairs of the plateau. These tasks are quite general, and provide a good base for the test of the Mobile Robot. These items generallly cover the majority if aspects of the obstacle course, but more information can be found on [12].

Now that a general description of the environment in which the MR should operate has been established, it is necessary to convert it into a set of requirements, based on which the MR should be designed and constructed. As this thesis is produced within a limited timeframe, a set of delimitations will also be introduced. This leads to the next chapter, which describes the requirements specification and delimitations.

Chapter 3

Requirements and delimitations

To make a more detailed description of the robotic system developed in this thesis, a set of requirements to the system is set up, based on the details given from the description of the operating environment. As this project is made under a limited timeframe 3.2 will describe a set of delimitations

3.1 Requirements

In the following section, a list of requirements is presented. These requirements have been extracted from the information regarding the obstacle course found in chapter 2. As these requirements concern various aspects of the mobile robot (MR), a division into several sub categories has been conducted.

3.1.1 Physical requirements

The first set of requirements of the mobile robot concerns the physical appearance and configuration.

- Dimensions
  From the description of the obstacle course, it is seen that the yellow gates are only 45$\pm$ 2cm wide, and that the tunnel is only 25 cm high. This, combined with the fact that the steps of the stair are only 40 cm long, provides the following maximum dimensions of the MR:

$$L \times W \times H = 40cm \times 43cm \times 25cm \tag{3.1}$$

- Static Clearance In the description of the obstacle course, it is given that area (9) is a hump track, with holes which are 2 cm deep, and humps which are up to 8 cm high. This entails that the MR should have a good clearance, and be able to walk over the humps, and climb an edge of 2 cm. See the dynamic requirements to clearance for more details.

3.1.2 Dynamical requirements

The second set of requirements concerns the dynamic capabilities of the MR.

- Linetracking
  To move around on the obstacle course, the MR needs to be able to track the black line. The MR should thus be capable of tracking a black line.
- Speed
  Speed is only important in two places on the obstacle course, when reaching the first gate, and when moving on the racetrack. The only requirement for a minimum speed is 0.25cm/s. This entails that the MR should move with a speed of at least 0.25m/s
- Radius of curvature
  The radius of curvature must not exceed 50 cm.
- Move up/down a slope.
  The two slopes should enable the MR to move up a slope of 9 degrees, and down a slope of 14 degrees.
- Clearance
  As described earlier, the MR should be able to climb a step of 2cm, and pass over obstacles of 8 cm. This can be reduced to two requirements of clearance, one from the ground to the body, and one from the ground to the endpoint of the legs, when they are moved.

$$C_{body} > 8[cm] \tag{3.2}$$

$$C_{Endpoint} > 2[cm] \tag{3.3}$$

- Performance
  From the introduction to this thesis, it was described that one of the greatest advantages

of legged locomotion over wheeled locomotion was the fact that it made movement in rough terrain possible. This came however at the cost of energy efficiency. As the obstacle course of the robocup competition geerally is composed of flat terrain, the MR will generally dispatch more energy than its wheeled counterparts. When generating the gait, this effect should be taken into account.

Note that the second requirement regarding radius of curvature is somewhat uninformative, as a hexapod should be able to turn in place.

3.1.3 Sensor processing requirements

The third set of requirements is related to the sensor input and its processing.

- Discern black,grey and white areas
  To find its way around the track, the MR needs to be able to track a black line over a white floor, and even on grey tiles.

- Robustness
  Even though the "white floor" of the competition is quite clean and free of obstacles, there are many scuff marks, and colour deviations. This entails that it is necessary for line tracking algorithm to be able to produce a reliable result, even when the floor is dirty.

- Crossroads
  As several crossroads are a part of the RoboCup track, it is necessary that the line tracking system is able to detect these and make the MR able to navigate in either direction.

- The linetracking software
  As several of the assignments on the track require that the MR is able to detect various objects, golf balls, and artificial plants, the sensor system should be able to provide the MR with the ability to recognize simple objects.

3.1.4 Intelligence

The fourth section of requirements is related to the intelligence of the system, and the general software structure.

- Divided into subsystems
  To make the software system of the MR as easy to maintain as possible, the system should be made in a modular fashion, so that subsystems can be exchanged.

- Map
  As a map of the obstacle course is always made public before the competition, it would only be sensible, if the MR were able to utilise this information, when determining how the MR should move

- Route
  When a route is given to the MR, or if the route through the environment needs to be changed, the map of the surroundings should be utilised to derive a path, so that no cumbersome reprogramming should be made.

- Pathfinding
  If the MR discovers that it is at a wrong location, it should be able to find a new route to the desired location by utilising the map information.

- Moving outside of line
  As the obstacle course is divided in two, with no guidance track between them, the software system should provide the MR with the ability to move outside of the black track.

3.2 Delimitations

As the master thesis is produced within a limited timeframe, it is necessary to determine exactly what parts of the MR will be analysed and implemented.

3.2.1 Localization

When designing a MR system, it is of great importance to derive the exact position of the MR. This can be done in various ways, ranging from using a GPS system, to using dead reckoning. In this thesis a localization algorithm will not be implemented. This entails that the developed system for path finding, mentioned in 3.1.4 can only be fully utilized if such an algorithm was implemented.

3.2.2 Sensor processing

As described in 2, several of the assignments of the obstacle course requires the ability to recognise simple shapes. To reduce the workload only an algorithm for line tracking will be implemented. This algorithm should however be expandable in such a way, that it provides the foundation for further processing.

Now that a set of requirements and delimitations have been derived, a basis on which the MR should be designed has been established. In the following section the MR platform will be introduced.

Chapter 4

Platform description

In the following chapter, the platform of the hexapod robot will be described. It is generally devised in such a way, that it is able to comply with the requirements given in section 3. This chapter will be divided into three different parts, each describing one area of the MR.

- **The physical platform**
  The physical platform consists of the metallic plates that compose the hexapod robot. It is this platform that defines the kinematic description of the robot.

- **The electric platform**
  The electric platform consists of three main components, **The sensory platform**, which is the platform that enables the robot to sense the environment. **The actuator platform** is composed of the motors, which enables the robot to move, and the motor drivers. This platform defines the dynamic properties of the system. Finally the **microcontroller and other electronics**, is the last element of the electronic platform. This platform interfaces the Sensory platform and the Actuator platform. As this element also contains the microcontroller it is by means of this element that the developed control strategy should be implemented.

- **Software development**
  The software development platform
  Whereas the other two elements are oriented against the physical capabilities of the robot, this is generally more oriented against the interface to a PC, and is as such directly connected to the microcontroller described in the electronic platform.

4.1 Physical platform

The physical platform is the physical appearance of the robot. This is the platform which defines the kinematic properties of the robot. The physical platform is generally based on a prefabricated Hexapod robot which has been bought from Lynxmotion [13] called AH3-R Walking Robot, which can be seen from Figure 4.1 This robotic platform supports six legs,

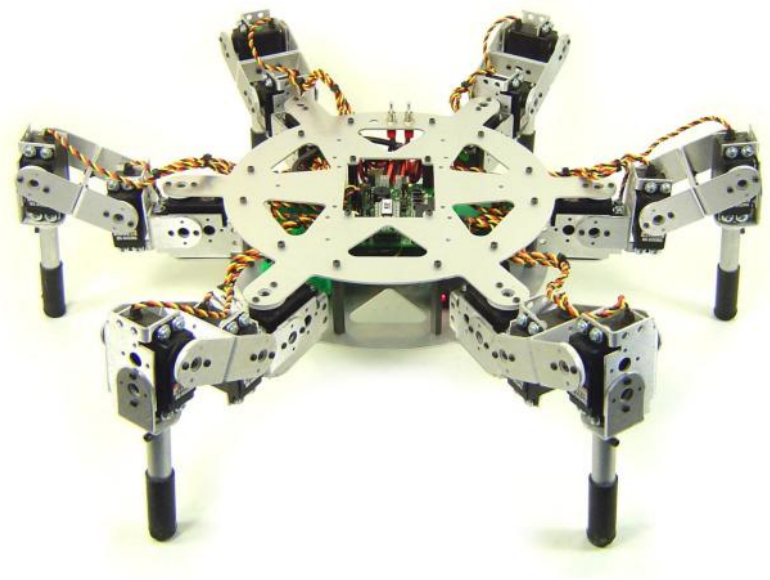

Figure 4.1: The overall appearance of the hexapod robot with servo motors

each with three degrees of freedom. According to [13], the dimensions can be found from table 4.1

dimension	value[cm]
Height (body)	5
Height (overall)	17.2
Width (body)	21.25
Width (overall)	45
Ground clearance	up to 11.6

Table 4.1: Shows the dimensions of the hexapod robot found from Lynx motion[13]

From this table it can be seen that the MR is a bit too wide to pass through the gates of the RoboCup competition. This entails that a new body should be constructed. This new body is constructed by two parallel layers of aluminium in the same way as the original body. These new aluminium parts can be seen from figure 4.2 and 4.3. This body changes only the width of the MR, which in stance phase now measures about 35 cm, which is small enough for

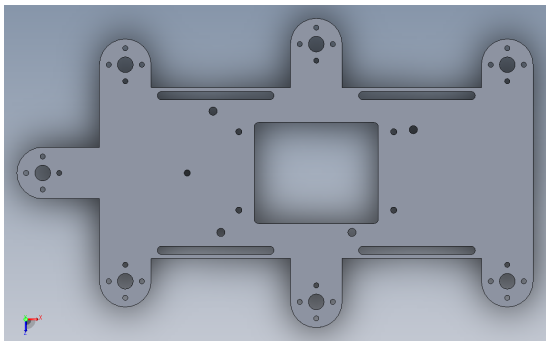

Figure 4.2: Shows the top layer of the new aluminum body

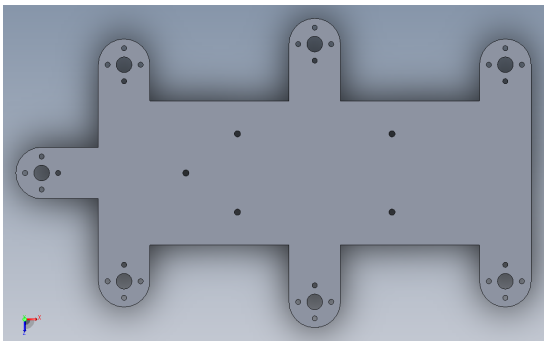

Figure 4.3: Shows the bottom layer of the new aluminum body

the MR to pass through the gates, and at the same time providing adequate space for the electronics. A 3d rendering of the hexapod robot can be seen from figure 4.4, This 3d model is used when making the dynamic model, and can be found in section B.

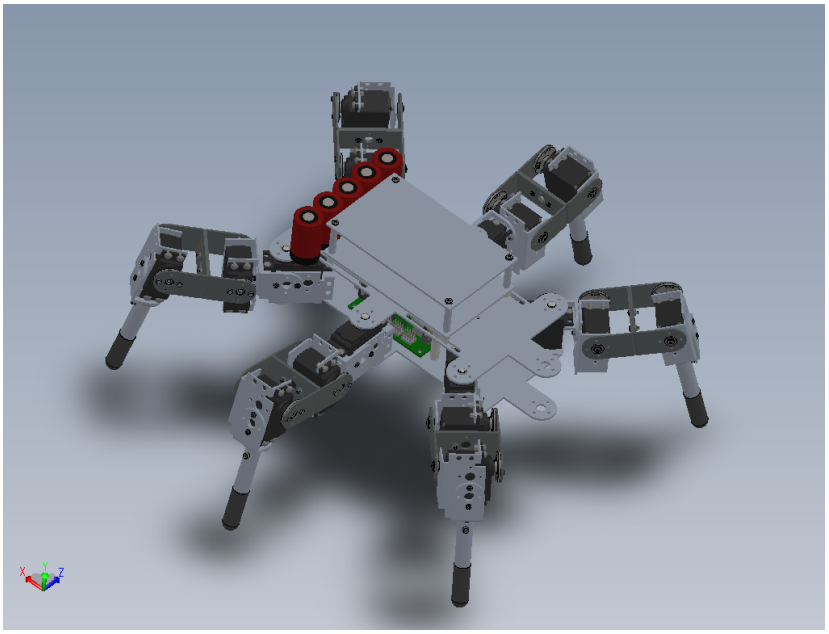

Figure 4.4: Shows a 3d rendering of the hexapod robot, with the new body

4.2 Electronical platform

The following section describes the electrical components of the mobile robot. It is divided into three sections, one describing the actuators of the MR, one describing the sensors of the MR, and one describing the On board computer and the interfaces to the sensors and actuators.

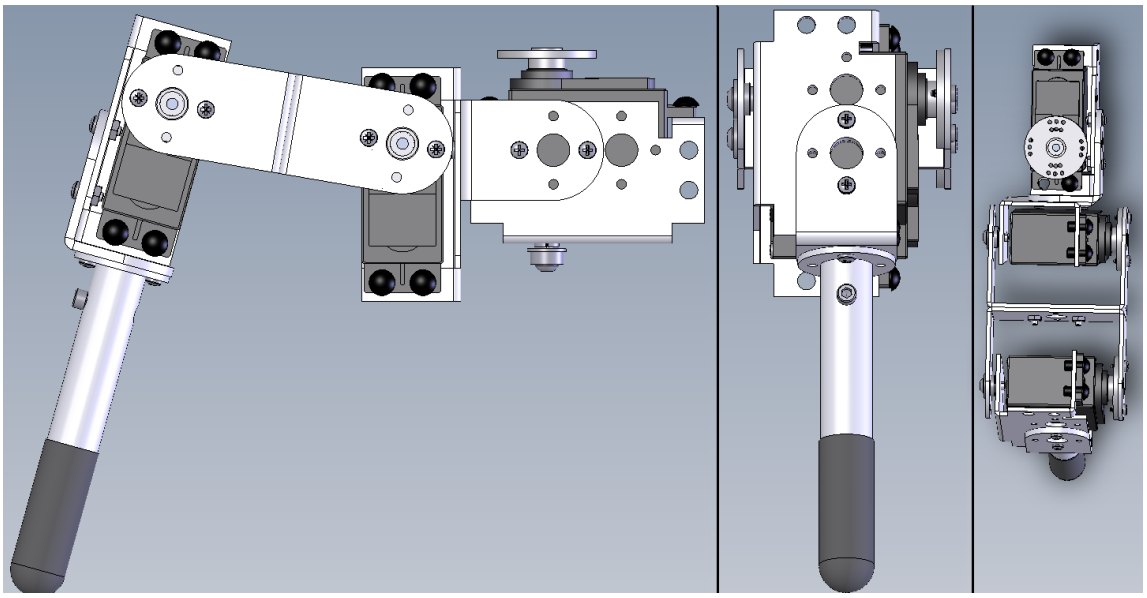

Figure 4.5: Shows an image of one of the legs of the MR

4.2.1 Actuators

The part of the electrical platform which composes the actuators is composed by two main elements, the servo motors, and an interface/driver print.

Servo motors

These servomotors control the angles of the joints of each leg, and thus provide propulsion for the MR. Figure 4.5 shows a sideways, frontal and top view of one of the legs, and the configuration of the servo motors. As each leg is manipulated by three servo motors, it is necessary to develop a naming convention which describes the individual joints of the robot leg. There are several "standard" conventions regarding names of such parts, one can be found from [14, p89-91], but is more related to industrial manipulators. In this study it is done by an analogy to the biological world, which has been used in the literature of hexapod robots[15] and [16, Excluding the trochanter which is a consequence of this article working with 4 DOF on each leg]. The first joint is called "Coxa" from the Latin word for hip, the second called "Femur" from the Latin word for thigh bone and finally "Tibia" from the Latin word for the shinbone.

The servo motors are internally controlled by a position feedback which entails that no information regarding the orientation of the servos is provided to the outside of the servo.

All servos of the MR are identical, and are manufactured by Hitec, and are of the type HS-645MG. The specifications of the motors can be found in [17]. It is seen that the servo motors

Specification	Value
Torque 4.8V/6.0V	0.75Nm / 0.94[Nm]
Speed 4.8V/6.0V	4.3[rad/s] / 5.23[rad/s]
Size	40.60 x 19.80 x 37.30mm
Weight	55.2g
Bearing Type	Dual Ball Bearing
Gears	Metal

Table 4.2: Shows the servos used in the Femur and Tibia joints. It is noticed that the servos are of "standard" size and are of the type HS-645MG

are of the type used in most radio controlled cars etc, although in high quality.

Servo interface print

As described earlier, there are 18 servo motors on the MR. These servo motors are controlled by means of a three pin connector. This connector includes one power connector, one ground, and one control signal. The control signal controls the position of the servo motor by means of a pulse length. This pulse ise sent with a frequency of 50Hz, and can vary in duration from approximately 800 - 2200μs. This entails that either the On board computer should use 18 ports to control the servo motors or an interface board should be used. As the most practical solution is to use an interface board, this solution is chosen. The chosen board is a ssc-32, from lynx motion[18]. The SSC-32 is capable of controlling 32 servos, with a RS232 interface. By sending a command as:

```
"#5 P1600 <cr>"
```

This command tells the interface print to keep sending impulses with a width of 1600 μ s to servo number 5. Every statement is ended by <cr> carriage return. If more servos are to be updated, one can combine statements, and end the whole line by <cr>, like:

```
"#5 P1600 #6 P700 <cr>"
```

This command will update both servo 5 and 6.

4.2.2 Sensory platform

The next part that should be considered is the sensor input to the MR. These sensors should be able to provide the MR with the required prerequisites to be able to comply with the requirements described in section 3, regarding the sensor input and their processing. To comply with the line tracking capabilities of the MR it has been chosen to install a camera on the MR, as it is able to provide huge amounts of information regarding the environment, and thus provides a broad basis, on which robust line tracking algorithms can be implemented. Beside the camera, pressure sensors on the end points of the MR legs have been mounted.

Pressure sensors

The pressure sensors of the MR are Force Sensing Resistors (FSR) mounted on the endpoints of the legs, underneath a rubber shield. The manufacturer[13] does however not provide the specific type/name of the resistor, but links to a document describing the 4000 series from interlink electronics[19].
As recommended by the assembly guide, the FSR is put in series with a 10 kΩ resistor, to generate a usable output. It does however quickly become clear that the force sensor cannot be made to work properly. When the endpoints of the MR legs strike the ground, the rubber shield deforms, and pressure is applied to the FSR. When the endpoints leave the ground, this deformation should be removed, and the pressure on the FSR should decrease to zero. The rubber shield does however have a tendency to keep its deformation, and thus keeps a constant pressure on the FSR, even when the endpoints have left the ground. This entails that these sensors will not be used in the further study of this report, except for a small test conducted in section 21.

Camera

The second sensor implemented on the MR is the camera. The camera mounted on the MR is the CMUcam3 [20], which is an open source camera, developed at the Carnegie Mellon University in Pittsburgh in 2007. It consists of a simple CMOS optical sensor, coupled with a 60 MHz ARM processor. This entails that primary image processing can be carried out directly on the camera, and then transferred to the main processor for further processing. It is interfaced by an RS232 connection, using either TTL or line levels. A further description of the capabilities of the CMUcam3 will be given in section 12.

4.2.3 On Board Computer and Interfaces

Now that the sensor and actuators of the system have been defined, it is time to describe where the obtained data will be processed.

On Board Computer

To provide a foundation for an intelligent autonomous system, it has been chosen to use an On board computer. The On board computer used for this project is the TS-7400 from Techno- logic Systems, [21], which can be seen from Figure 4.6. Some of the features which

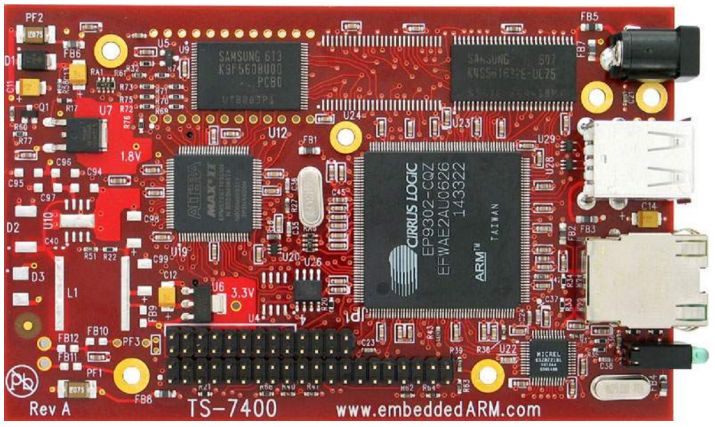

Figure 4.6: Shows the on board computer used on the MR

have entailed that this board has been utilised are.

Knowledge

As the computer has been used in other project at the university, knowledge about interfaces and driver code is readily available.

Speed

The TS-7400 is based on a 200 MHz ARM9 processor with integrated math engine. This math engine speeds up floating point calculations, as it is supported in hardware and not emulated.

On-board 1/10/100 Mbit LAN

A standard LAN connector is located on the board, which provides the user with a simple means of communicating with the board, via ssh.

USB

There are two USB 2.0 interfaces, one of which a former group has utilised to connect a wireless USB dongle [22], which can provide a wireless connection between A PC and the

mobile robot.

SD-card
The board comes with a SD-card socket. To simplify the development of the embedded system, one can install a Linux distribution on this card, from which the system can boot. This provides the user with the option of connecting to the board via ssh or similar strategies. This has been done by [22], who has installed a complete debian distribution on a 1GB SD-card.

TTL level UARTs
There are three TTL level UARTs on the board. Two of which will be used to interface other hardware components.

A/D converters
The board provides four 12-bit A/D converters.

Digital I/O
The board provides 20 digital inputs/outputs

Size
The board measures only 7.4 cm $\times$ 12 cm, which entails that it can fit on the robot

Power
The board needs a standard 5V regulated power supply, and at 200 MHz it uses up to 450mA. [23]

Another big advantage of the TS-7400 is the fact that technological systems provide a wide range of code examples, test programs, Linux distributions and well documented manuals [24].

Hardware interfaces

Now that the main hardware components of the MR have been determined, it is time to see how these are interconnected. The main hardware component of the MR is the On board computer. As both the CMUcam3 and the SSC-32 are interfaced by RS-232 connections, it is clear that two of the three RS-232 TTL connections from the board should be used. Thus the hardware model becomes quite simple, as only four main components are present (not including batteries and switches): the servos, the servo controller, the on board computer and

the CMUcam3. These are linked together as seen on figure 4.7.

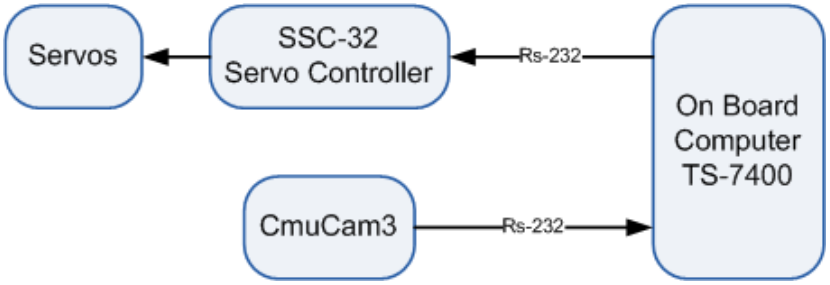

Figure 4.7: Shows a simple model of how the four main hardware components are connected

4.3 Software development

As described in 4.2, there are two microprocessors on the MR. The first one is the On board computer, the ts-7400, and the other one is the CMUcam3. Code for both microprocessors are developed under the CYGWIN Linux environment under Windows.
The code for the Ts-7400 is compiled directly on the on board computer to that the correct compiler is used. The code for the CMUcam3 is developed under what is called the "virtual-cam"[25]. This is a development environment that emulates the CMUcam3 hardware, where images are supplied by simple image files. This environment gives more warnings and error descriptions, than when running the program directly on the embedded hardware, and thus speeds up the process of code development and debugging.

Now that a description of the mechanical, electrical and software aspects of the robot has been made, it is time to consider the more theoretical mathematical aspects. This will be done in the next part of the report, which concerns the mathematical modelling of the MR.

Part conclusion

In the previous part of the report, an introduction to the general concepts of the benefits of legged locomotion were given. Then a description of the operating environment in which the MR is to operate were presented. This environment were based on the track of a competition for Mobile robots, called Robocup. From this environmental description, a requirements specification were derived, which again led to a set of delimitations. From these requirements a mobile robot were built. This MR were partly based on a prefabricated robot from LynxMotion, but a new body were constructed to make it fit through the obstacles in the environment. Hardware were chosen for this robot, which should make it able to fulfill the found requirements.

Now that a description of the mechanical and electrical and software aspects of the robot has been made, it is time to consider the more theoretical mathematical aspects. This will be done in the next part of of the report, which concerns the mathematical modelling of the MR.

Part II

Modelling

Introduction

The following part of the report is devoted to the modelling of the hexapod robot. This derivation of the model is conducted in three steps. These three steps are all different parts of the entire system model, which will be used to control the position of the robot.

1. Forward kinematic robot model
   The forward kinematic model serves the purpose of being able to determine the position of the robot by means of the angles provided by the servo motors of the legs. This model is used to determine the rotations needed to transform coordinates between the various frames of the system.

2. Inverse kinematic robot model
   The inverse kinematic model is a model describing which angles should be used if a certain position of the end effector is needed. These positions, when combined in the correct way, yield a gait of the hexapod robot.

3. Dynamic robot model
   Finally, the dynamic robot model is the model describing the forces and torques acting on the robot, and servo motors.

In general, what the model should be able to provide is a description of how the robot will act when certain position inputs are applied. The three sub models described above are connected in the way shown by 4.8. The structure of the model should be understood in

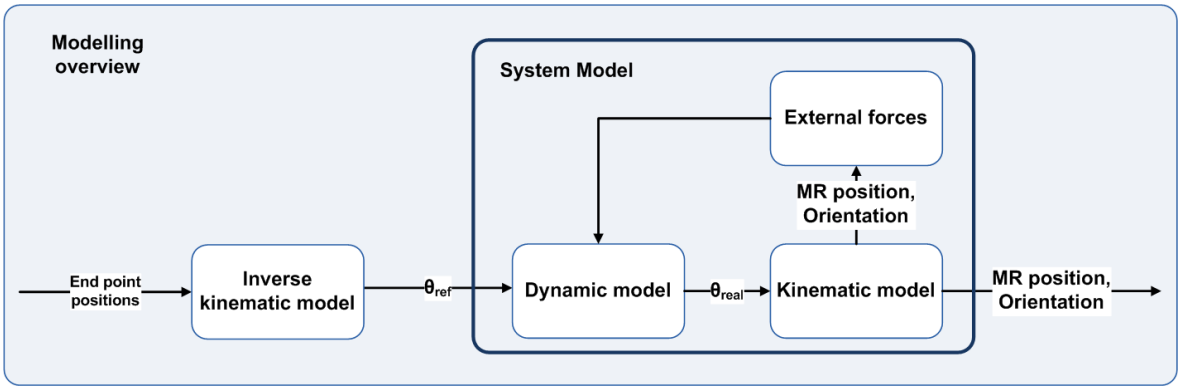

Figure 4.8: The structure of the robot model, shown with three sub models, the inverse kinematic, the dynamic and the forward kinematic. The angles θ should be understood as all angles of the robotic legs. Notice the distinction between the real system and the software running on the robot.

the following way. A position to the end positions of the legs is given. Then the inverse kinematic model describes the angular positions of the legs of the robot to achieve the given position. These angular positions are used as a reference for the dynamic model, and the

dynamic model then describes how the real angles change with the reference angles. Finally the forward kinematic model transforms these angles to the position of the robot. Depending on the configuration of the legs, the position of the legs of the MR determines how it interacts with the environment. If the legs touch the ground, a ground reaction force occurs, which again affects the dynamic model. In the following chapters, the derivation of the kinematic model is conducted first, then the inverse kinematic model, and then finally the dynamic model. After the whole model has been derived, it is used to give an interpretation of the stability of the hexapod robot. In the final sections of this part of the report, these models are verified.

Chapter 5

Direct kinematic model

The first part of the model that should be made is the kinematic model. When regarding kinematics, one does only treat the motion of the system without regard to system forces. That is, kinematics describe the position/motion of the robot with regard to the position/motion of the actuators of the system, and not the forces they imply on the robot. In section 4.2.1 it was defined that each leg of the robot consists of three degrees of freedom. The joints of these degrees of freedom were labelled respectively Coxa, Femur and Tibia. In the following a more thorough description of these limbs is presented, which leads to the derivation of the kinematic model of each limb, and by combining these, the kinematic model of the robot. To be able to develop a kinematic model, the first thing to do is to define the coordinate systems which are included in the model.

5.1 Defining coordinate systems

The first part of the kinematic model which needs to be derived is the various coordinate systems which are used in the model. One might claim that only a single coordinate system is required, namely the global frame in which the robot operates. However, this would entail that the derived kinematic equations would become hard and cumbersome to derive. Instead a multiple coordinate frame approach is used. In the following section these coordinate frames are defined.

5.1.1 The global coordinate system

To be able to model the system, it is necessary to determine a global coordinate system within it should operate. This coordinate system is chosen to have it's origin in the lover right of the obstacle course, with a positive x axis parallel and next to the wall, and with an y axis perpendicular to this, with a positive axis pointing in the left direction if seen from the origin in the direction of the x axis. The z axis is pointing directly upwards. All movements of the robot are going to be with this as a reference. The chosen coordinate system can be seen from figure 5.1. The next thing that should be defined is the position and orientation of the

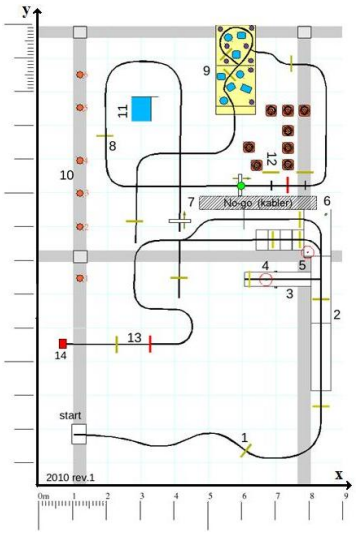

Figure 5.1: Shows the chosen global coordinate system when applied to the RoboCup obstacle course

robot in this coordinate system.

5.1.2 Definition of robot coordinate system

To define the orientation and position of the robot, another coordinate frame should be attached directly to the robot. The origin of this coordinate frame is attached to the centre, between where the two middle legs are attached to the body of the MR, with a x axis pointing in the direction of the head, and the y axis pointing to the "left", and with a z axis perpendicular to these two. This attached coordinate frame can schematically be seen from figure 5.2

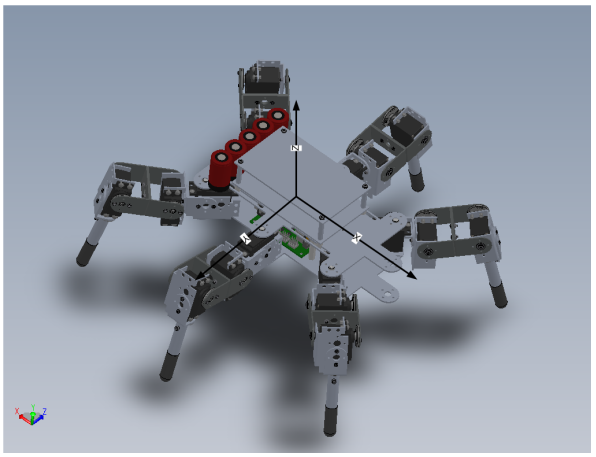

Figure 5.2: Shows the attached coordinate frame attached to the robot

5.1.3 Defining the leg coordinate frames

The derivation of the coordinate frames of the robotic legs takes its beginning in the description of the legs as several individual bodies, each with one degree of freedom. These bodies are called "links". These links are interconnected and thus form (long) chains of bodies. Due to the three degrees of freedom of the legs one should divide them into three links, each with one degree of freedom. This process can be seen from figure 5.3. To be able to describe this

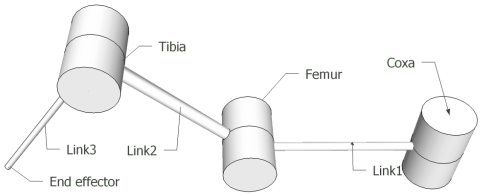

Figure 5.3: Shows the links of a robotic leg attached to the three joints Coxa, Femur and Tibia. The cylinders in the drawing show the direction of the rotation.

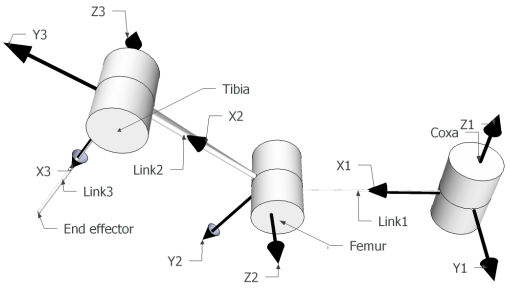

Figure 5.4: Shows the links of a robotic leg with attached coordinate systems as per the Denavit Hartenberg notation.

system adequately, a total of four coordinate frames should be attached. These are named frame {0}-{3} or "leg", "Coxa", "Femur", and "Tibia" frames. Frame {0} is attached to the point where the leg is attached to the mechanical platform, with a z axis pointing upwards, the x axis pointing out from the body, and the y axis finalizing a right hand coordinate system. The remaining three coordinate systems are attached after a procedure described in [14, p77]. In this procedure, a coordinate system is attached to the three links in the following way:

1. The z_i axis should be defined along the axis which joint *i* rotates about.

2. The x_i axis points in the direction from joint *i* to *i+1*.

3. The y_i axis should be assigned so that a right hand coordinate system is achieved.

4. Choose the origin as to make as many parameters zero as possible. This entails that the origin of frame i is located where link length a_i intersects joint axis i.

By using this procedure the coordinate frames seen from figure 5.4 are found. It should be noted that frame $\{0\}$ and $\{1\}$ are perfectly aligned when the angle of the Coxa joint equals zero.

5.2 Moving from coordinate frame to coordinate frame

Now several coordinate frames have been defined. But to derive the kinematic model it is necessary to describe how these frames are interconnected, and how one describes points in one coordinate system by means of another. In the following, three cases are considered.

1. Moving from robot coordinates to global coordinates

2. Moving from leg coordinates to robot coordinates

3. Moving from link to link in leg coordinates

These three cases are considered in the following sections, but first some general definitions are considered

5.2.1 General considerations

By giving a reference coordinate system $\mathcal{A}$ any link frame $\mathcal{L}$ can be described by a position vector ${}^{\mathcal{A}}\widehat{\mathcal{L}} = [x_{\mathcal{A}}, y_{\mathcal{A}}, z_{\mathcal{A}}]$, this should be read, the position vector of the link $\widehat{\mathcal{L}}$ defined by the coordinate system $\mathcal{A}$. To define an orientation of the link, a coordinate system is attached to the link called $\mathcal{B}$. One can describe the transformation from $\mathcal{A}$ to $\mathcal{B}$ by means of a transformation matrix ${}^{\mathcal{A}}_{\mathcal{B}}\mathcal{T}_{\mathcal{R}}$, which describes the rotation in space. This transformation matrix is called the direct cosine matrix or DCM. The DCM can however only describe the transformation if the origin of the two coordinate systems is coinciding. This entails the following relationship:

$$ {}^{\mathcal{A}}\widehat{\mathcal{L}} = {}^{\mathcal{A}}_{\mathcal{B}}\mathcal{T}_{\mathcal{R}} {}^{\mathcal{B}}\widehat{\mathcal{L}} \tag{5.1} $$

To describe the more general case where the origins do not coincide, the following equation should be considered, as this includes distance from $\mathcal{A}$ to $\mathcal{B}$ by means of the vector $\widehat{\mathcal{L}}_{\mathcal{A}\to\mathcal{B}}$

$$^{\mathcal{A}}\widehat{\mathcal{L}} = {}^{\mathcal{A}}_{\mathcal{B}}\mathcal{T}_{\mathcal{R}} \cdot {}^{\mathcal{B}}\widehat{\mathcal{L}} + \widehat{\mathcal{L}}_{\mathcal{A}\to\mathcal{B}} \tag{5.2}$$

That is, the new position is a function of the rotation of the coordinate system plus the vector describing the displacement of the origins. To ease this notation, this is put into matrix form, and the following is obtained:

$$\begin{bmatrix} {}^{\mathcal{A}}\widehat{\mathcal{L}} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{\mathcal{A}}_{\mathcal{B}}\mathcal{T}_{\mathcal{R}} & \widehat{\mathcal{L}}_{\mathcal{A}\to\mathcal{B}} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^{\mathcal{B}}\widehat{\mathcal{L}} \\ 1 \end{bmatrix} \Leftrightarrow \tag{5.3}$$

$$^{\mathcal{A}}\widehat{\mathcal{L}} = {}^{\mathcal{A}}_{\mathcal{B}}\mathcal{T} \cdot {}^{\mathcal{B}}\widehat{\mathcal{L}} \tag{5.4}$$

Where in the latter notation, it is not described that the position vectors in fact do contain the number 1. The problem is now reduced to finding the transformation matrix ${}^{\mathcal{A}}_{\mathcal{B}}\mathcal{T}$. This transformation describes the transformation from one coordinate system to another. If more than two joints are joined by links it is possible to extend the above definition by means of (left) multiplication:

$${}^{m}_{n}\mathcal{T} = \prod_{i=n+1}^{m} {}^{i}_{i-1}\mathcal{T} \tag{5.5}$$

That is, it is possible to describe the position of any point in one of the frames attached to any link, by means of a coordinate system defined by another link. To be able to do this, it is however necessary to define the chains of links describing the motion of the robot. This is done in the following. The approach to find these transformation matrices is somewhat different in the three cases. In the first two, moving from global frame to robotic frame, and then to leg frames, a direct approach is defined. But when considering moving from link to link in the legs, the Denavit Hartenberg notation is used, as it is easily applied.

5.2.2 Kinematics from leg frame to robot frame

The first and simplest case that is considered is moving from the leg frame or frame $\{0\}$ to the robot frame. To be able to derive the transformation matrix, one has consider what kinds of transformations are taking place when moving from the leg frames to the robot frame. The first thing that is noted is that they are all located in the same plane, that is, no translation in the z axis. The next thing that is noted is that only a rotation around the z axis is present. In [14, p372] several rotational matrixes are defined, and a rotation around the z axis can be

found to be:

$$ {}^{\mathcal{A}}_{\mathcal{B}}\mathcal{T}_{\mathcal{R}} = \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.6} $$

As the translation remains constant, a complete description can now be achieved by using 5.4. As there are 6 different leg frames, six different transformation matrices should be defined. The names of the matrices are derived from the naming of the robotic legs made in 4.2.1, and can be seen from table 5.1 along with the parameters needed to construct these matrices. The values are found from the constructed 3D model described in section B. 8u One should note

Leg	Name	x[cm]	y[cm]	ϕ[Degrees]
Left 1	${}^{P}_{L1}\mathcal{T}$	9.25	5.25	45
Left 2	${}^{P}_{L2}\mathcal{T}$	0	6.25	90
Left 3	${}^{P}_{L3}\mathcal{T}$	-9.25	5.25	135
Right 1	${}^{P}_{R1}\mathcal{T}$	9.25	-5.25	-45
Right 2	${}^{P}_{R2}\mathcal{T}$	0	-6.25	-90
Right 3	${}^{P}_{R3}\mathcal{T}$	-9.25	-5.25	-135

Table 5.1: Shows the naming and parameters of the derived transformation matrices

that the angles presented in table 5.1 describe the rotation of the robot coordinate system to point in the direction of the various leg frames. This entails that six transformational matrices have been derived which makes it possible to describe the leg frames by means of the robot coordinate frame. The derived matrices can be found in appendix A. The next thing that should be considered is how to move from the robot frame to the global frame.

5.2.3 Kinematics from robot frame to global frame

As before, this transformation is based on geometrical interpretations, and not on the Denavit Hartenberg notation. To be able to describe the robot position in global coordinates, one has to assume that the robot coordinate system could have any orientation and bias compared to the global coordinate system. That is, the robot can have three different rotation angles, the pitch, yaw and roll, but also a bias of X,Y,Z. To align the orientation of the robot, it is necessary to rotate the coordinate system in all these three directions. To be able to do this, three different DCM matrices are used, one for rotation along each axis (see [14, p372]):

$$ \mathcal{T}_{R^G_P} = \mathcal{T}_{R-yaw} \cdot \mathcal{T}_{R-roll}\mathcal{T}_{R-pitch} \tag{5.7} $$

These three are given by the following three equations:

$$\mathcal{T}_{R-yaw} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(yaw) & sin(yaw) \\ 0 & -sin(yaw) & cos(yaw) \end{bmatrix} \tag{5.8}$$

$$\mathcal{T}_{R-roll} = \begin{bmatrix} cos(roll) & 0 & -sin(roll) \\ 0 & 1 & 0 \\ sin(roll) & 0 & cos(roll) \end{bmatrix} \tag{5.9}$$

$$\mathcal{T}_{R-pitch} = \begin{bmatrix} cos(pitch) & sin(pitch) & 0 \\ -sin(pitch) & cos(pitch) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

When this is done, the bias should be included. This is done by adding the position of the robot (X,Y,Z).

$${}^{G}_{P}\mathcal{T} = \begin{bmatrix} T_{R^{G}_{P}} & \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \\ 0 \quad 0 & 1 \end{bmatrix} \tag{5.11}$$

Where Z is the height above ground level.

5.2.4 Kinematics from link to link

The last case which will be considered is moving from link to link in the robot legs. As already described, the legs are composed of three links, and thus three coordinate frames, which again entails that three transformation matrices should be found. These are found by means of the Denavit Hartenberg notation What the Denavit Hartenberg notation is all about, is to make a uniform way of describing a given chain of bodies/links connected by joints. In this sense, a link is in fact a connection of two joints, that is, link i connects joint *i-1* and *i*. According to the notation, the kinematics of each link can be described by means of four parameters[14, p74]; two that describe the link itself a_i and α_i, and two that describe its connection to the next link d_i and θ_i. These four parameters can be described in the following way[14, p76];

- a_i The distance from joint i-1 to i measured along their connecting axis, x_i
- α_i The angle between i-1 and i measured around their connecting axis, x_i
- d_i The distance from i-1 to i measured along their perpendicular axis z_i

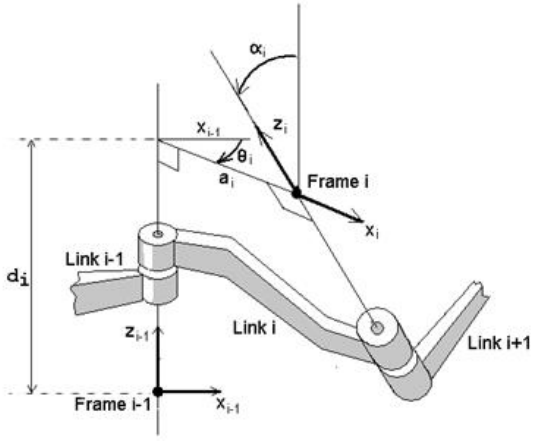

Figure 5.5: Shows the parameters used in the Denavit Hartenberg notation, when describing the transformation from one coordinate frame to another. The above figure is taken from [26]

- θ_i The angle between i-1 and i measured along their perpendicular axis z_i

A geometrical interpretation of these parameters can be seen from 5.5.

By using the coordinate frame derived in 5.1.3 and the described link parameters, it is possible to determine the transformation matrix by means of the following equation:

$$ {}^{i}_{i-1}\mathcal{T} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & a_{i-1} \\ sin(\theta_i) \cdot cos(\alpha_{i-1}) & cos(\theta_i) \cdot sin(\alpha_{i-1}) & -sin(\alpha_{i-1}) & -sin(\alpha_{i-1}) \cdot d_i \\ sin(\theta_i) \cdot cos(\alpha_{i-1}) & cos(\theta_i) \cdot sin(\alpha_{i-1}) & cos(\alpha_{i-1}) & cos(\alpha_{i-1}) \cdot d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.12} $$

In the above equation, only the angle θ changes with time, and is the parameter controlled by the position of the servo motors.

Applying the notation In section 5.1.1 it is described that the robot is moving in a global coordinate system, with a position described as being in the centre of the legs of the robot. To be able to conduct the appropriate analysis, a coordinate system is attached to the robot. This coordinate system has the centre of the legs as origin, the x axis pointing out of the robot head, the z axis being perpendicular pointing upwards and the y axis as finishing of the right-hand coordinate system. As the robot consists of 6 legs, each with 4 coordinate frames attached, it results in a total of 24 different coordinate frames. To be able to place the robot in the global coordinate system, two additional coordinate systems are needed, and thus 26 different transformation matrices are needed. Luckily the derivation of the transformation matrices for the legs are identical, as they are constructed in the same way. If *k* denotes the leg in question, the transfer matrices can be found by means of equation 5.13-5.15

$$
{}^{1}_{0}\mathcal{T} = \begin{bmatrix} cos(\theta_{1k}) & -sin(\theta_{1k}) & 0 & a_0 \\ sin(\theta_{1k}) \cdot cos(\alpha_0) & cos(\theta_{1k}) \cdot sin(\alpha_0) & -sin(\alpha_0) & -sin(\alpha_0) \cdot d_1 \\ sin(\theta_{1k}) \cdot cos(\alpha_0) & cos(\theta_{1k}) \cdot sin(\alpha_0) & cos(\alpha_0) & cos(\alpha_0) \cdot d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.13}
$$

$$
{}^{2}_{1}\mathcal{T} = \begin{bmatrix} cos(\theta_{2k}) & -sin(\theta_{2k}) & 0 & a_1 \\ sin(\theta_{2k}) \cdot cos(\alpha_1) & cos(\theta_{2k}) \cdot sin(\alpha_1) & -sin(\alpha_1) & -sin(\alpha_1) \cdot d_2 \\ sin(\theta_{2k}) \cdot cos(\alpha_1) & cos(\theta_{2k}) \cdot sin(\alpha_1) & cos(\alpha_1) & cos(\alpha_1) \cdot d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.14}
$$

$$
{}^{3}_{2}\mathcal{T} = \begin{bmatrix} cos(\theta_{3k}) & -sin(\theta_{3k}) & 0 & a_2 \\ sin(\theta_{3k}) \cdot cos(\alpha_2) & cos(\theta_{3k}) \cdot sin(\alpha_2) & -sin(\alpha_2) & -sin(\alpha_2) \cdot d_3 \\ sin(\theta_{3k}) \cdot cos(\alpha_2) & cos(\theta_{3k}) \cdot sin(\alpha_2) & cos(\alpha_2) & cos(\alpha_2) \cdot d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.15}
$$

As there are two different types of legs, a set of two different Denavitt Hartenberg parameters is found. One for the middle legs, and one for the front and rear legs. The parameters for the front or rear legs can be seen in table 5.2 and for a middle leg in table 5.3 By using these

Link	θ_i	α[degrees]	a[cm]	d[cm]
Coxa	θ_1	0	0	0
Femur	θ_2	90	3.8	-1
Tibia	$\theta_3 + 90$	180	5.5	0

Table 5.2: Shows the Denavit Hartenberg parameters of a robotic rear or front leg

Link	θ_i	α[degrees]	a[cm]	d[cm]
Coxa	θ_1	180	0	0
Femur	θ_2	90	5.4	-1
Tibia	$\theta_3 + 90$	180	5.5	0

Table 5.3: Shows the Denavit Hartenberg parameters of a robotic rear or front leg

three transformation matrices, it can be seen that the end effector is given as a coordinate in the Tibia frame. As the tibia is 14.5 cm long, the end effector is located at x=14.5cm, as the Tibia coordinate frame is assigned with the x axis pointing in the way of the connecting link. By using 5.5 these three transformations lead to the final position of the robot leg, defined from the leg frame.

$$
{}^{3}_{0}\mathcal{T} = {}^{3}_{2}\mathcal{T}\,{}^{2}_{1}\mathcal{T}\,{}^{1}_{0}\mathcal{T} \tag{5.16}
$$

With this equation, all three cases of coordinate transformations have been described, and it is now possible to describe any point in any of the coordinate frames of the robot in global coordinates. The position of a point in link frame {3} can be described by means of the following transformation:

$$ {}^{G}_{0}\mathcal{T} = {}^{G}_{P}\mathcal{T} \cdot {}^{P}_{leg}\mathcal{T} \cdot {}^{3}_{0}\mathcal{T} \tag{5.17} $$

where
${}^{P}_{leg}\mathcal{T}$ can be any of the transformation matrices given in table 5.1.
It should now be checked if the derived kinematic model actually is able to describe the robot.

5.3 Simulating the kinematic model

In the following section, a small simulation is conducted to verify the kinematic model. The simulation is done in such a way that a small sketch of the robot is made by means of the derived transformation matrices. All origins of the defined coordinate systems and the position of the end effector are transformed to global coordinates. The figure can be seen from 5.6

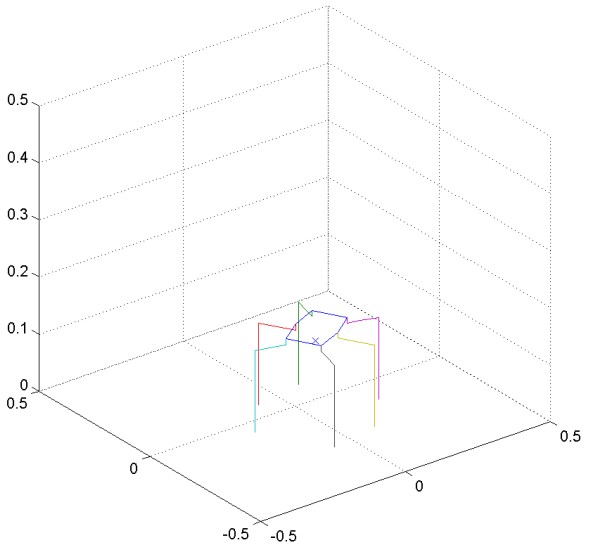

Figure 5.6: Shows the kinematic model of the MR, when it is plotted in simulink

This figure shows the MR situated in the position [x=0,y=0.2,z=15], with a rotation around the z axis of π/6 radians. It can be seen, that the result agrees with the schematical drawings of appendix B. It is thus concluded that the kinematic model is able to transform local coordinates in any of the links of the MR to the global coordinate system.

This leads to the next chapter of this thesis, which concerns the opposite problem. The problem of deriving angles, based on positions - the inverse kinematic model.

Chapter 6

Inverse kinematic model

This chapter is devoted to solving the problem of inverse kinematics of the hexapod robot. The inverse kinematic model can be said to solve the complete opposite problem of the forward kinematic model. Instead of being able to describe the position of the end point of the legs as a function of the angles of the servo motors, it should be possible to determine the angles of the servo motors by means of a given end point position. That is, finding the angles θ_1, θ_2 and θ_3 when a coordinate vector $P_e = [x_e y_e z_e]$ for the end effector is provided. This end point position can be given in global coordinates as well as leg coordinates. As the model is easily derived in local leg coordinates, a transformation from global to local coordinates is needed. In section 5.2.2 and 5.2.3 the transformations from first leg frame to robot frame, and then to global frame were developed by multiplying equation 5.11 with one of the transformations from table 5.1. The following transformation from leg to global coordinates is found.

$$ {}^{G}_{L1}\mathcal{T} = {}^{P}_{Lk}\mathcal{T} \cdot {}^{G}_{P}\mathcal{T} \tag{6.1} $$

To come from global coordinates to leg coordinates, ${}^{L1}_{G}\mathcal{T}$ is needed. According to [14, p45] the relation between these two transformations can be found by inversion, that is:

$$ {}^{Lk}_{G}\mathcal{T} = \left({}^{G}_{Lk}\mathcal{T}\right)^{-1} \tag{6.2} $$

This entails that the problem of inverse kinematics now can be solved in the same way whether global or leg coordinates are supplied.

6.1 Method

In [14, p109] two different methods for finding the inverse kinematic of manipulators are discussed, an algebraic, and a geometric. According to the book, the inverse kinematics of

a manipulator where the angles of α are either 0 or ± 90 degrees, the inverse kinematics can be derived quite easily. From table 5.2 and 5.3 in section 5.2.4 it is seen that all α fulfil this criteria. This entails, that the inverse kinematic model is based on simple geometrical considerations, and divided into two steps. The first step concerns the description of the angle of the Coxa joint; the second with the derivation of the Femur and Tibia.

6.2 The angle of the Coxa joint

The derivation of the first angle θ_1 is derived from figure 6.1, which shows the robotic leg if it is viewed from above. From this it is seen that the Femur and Tibia links form a straight line with the combined length ℓ. The fact that the two servos operate in the same plane of motion is a consequence of the fact that the α value given in 5.2 is equal to zero. By using

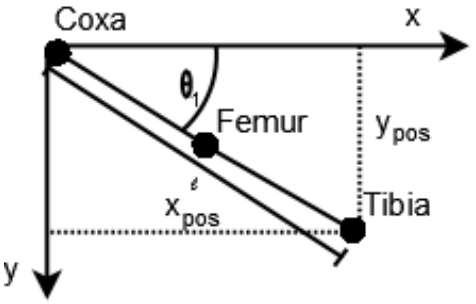

Figure 6.1: Shows the robotic leg as seen from above, with dots representing the Coxa, Femur and Tibia joints

simple geometric formulas, it is seen that the following relations are valid:

$$x_{pos} = cos(\theta_1) \cdot \ell \tag{6.3}$$

$$y_{pos} = sin(\theta_1) \cdot \ell \tag{6.4}$$

$$\theta_1 = atan\left(\frac{y_{pos}}{x_{pos}}\right) \tag{6.5}$$

This equation does however have its limitations. Equation 6.5 should be solved, in such a way that the original orientation of the leg is kept intact. This could be done using ATAN2(x,y) in MATLAB. Another thing that is noted is that if an endpoint underneath the robot body is provided, the equation will yield an angle which will make the leg swing into the robot body. The last thing that is seen is, that if a point that lies directly underneath the Coxa joint is provided, the equation can be satisfied by all angles. These special considerations can be quantified by means of looking at the x component of the reference position. If it equals 0, the last angle should be used, if is negative, 180 degrees are added. By taking these special

considerations into account, 6.5 is rewritten to 6.6

$$\theta_1 = \begin{cases} \theta_1 & x = 0 \\ atan2\left(\frac{y_{pos}}{x_{pos}}\right) + 180 & for \ x < 0 \\ atan2\left(\frac{y_{pos}}{x_{pos}}\right) & x > 0 \end{cases} \tag{6.6}$$

6.3 The angle of the Femur and Tibia joints

To derive the angle of the Femur and Tibia joints, another transformation is needed. This transformation should transform the coordinates from the leg frame to the area in the Coxa frame, where the Femur joint is. This transformation is conducted in a similar way as done in 6.2 when transforming from global to leg coordinates.

$$^0_1\mathcal{T} = \left(^1_0\mathcal{T}\right)^{-1} \tag{6.7}$$

but with an additional transation along the x axis of the Coxa frame. This entails that the new coordinates can be found to be:

$$P_{new} = ^0_1\mathcal{T} \cdot P + P_{trans} \tag{6.8}$$

$$\tag{6.9}$$

A drawing of the robot leg as seen from perpendicular to the plane of motion of the Femur and Tibia joints, in the Coxa frame is made 6.2.

From this drawing, it is seen that two triangles are found. One in the bottom marked by blue, and one marked by green. The first thing that is noticed about this drawing, is the fact, that the blue triangle is right-angled. This entails that the angle, θ_4 can be found from the following equation:

$$\theta_4 = atan(\frac{z_{pos}}{x_{pos}}) \tag{6.10}$$

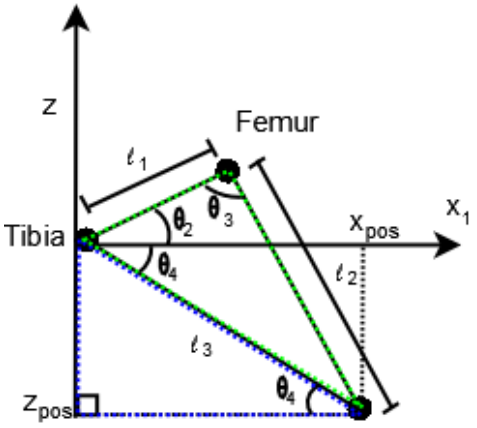

Figure 6.2: Shows a drawing of the leg, perpendicular to the x, z plane of the leg joint

The next thing, that is observed is the fact, that the two triangles share a common side, ℓ_3. As the blue triangle is right-angled, and the green one is not, two different expressions describe the length of ℓ_3. For the right-angled, one uses Pythagoras's theorem:

$$\ell_3^2 = x_{pos}^2 + z_{pos}^2 \tag{6.11}$$

And for the non right-angled one, the law of cosines is used. And by means of this, the two remaining angles are found:

$$\begin{aligned}
\ell_2^2 &= \ell_1^2 + \ell_3^2 - 2 \cdot \ell_1 \cdot \ell_3 \cdot cos(\theta_4 + \theta_2) && (6.12)\\
\frac{\ell_2^2 - \ell_1^2 - \ell_3^2}{2 \cdot \ell_1 \cdot \ell_3} &= -cos(\theta_4 + \theta_2) && (6.13)\\
\theta_2 &= acos(\frac{\ell_1^2 + x_{pos}^2 + z_{pos}^2 - \ell_2^2}{2 \cdot \ell_1 \cdot \sqrt{x_{pos}^2 + z_{pos}^2}}) - \theta_4 && (6.14)\\
\theta_2 &= acos(\frac{\ell_1^2 + x_{pos}^2 + z_{pos}^2 - \ell_2^2}{2 \cdot \ell_1 \cdot \sqrt{x_{pos}^2 + z_{pos}^2}}) - atan(\frac{z_{pos}}{x_{pos}}) && (6.15)
\end{aligned}$$

$$\begin{aligned}
\ell_3^2 &= \ell_1^2 + \ell_2^2 - 2 \cdot \ell_1 \cdot \ell_2 \cdot cos(\theta_3) && (6.16)\\
\frac{\ell_3^2 - \ell_1^2 - \ell_2^2}{2 \cdot \ell_1 \cdot \ell_2} &= -cos(\theta_3) && (6.17)\\
\theta_3 &= acos(\frac{\ell_1^2 + \ell_2^2 - x_{pos}^2 - z_{pos}^2}{2 \cdot \ell_1 \cdot \ell_2}) && (6.18)
\end{aligned}$$

This entails, that the three angles that define the position of the endpoint of each leg, can be described by means of the following three formulae:

$$\begin{aligned}
\theta_1 &= atan\left(\frac{z_{pos}}{x_{pos}}\right) && (6.19)\\
\theta_2 &= acos(\frac{\ell_1^2 + x_{pos}^2 + z_{pos}^2 - \ell_2^2}{2 \cdot \ell_1 \cdot \sqrt{x_{pos}^2 + z_{pos}^2}}) - atan(\frac{z_{pos}}{x_{pos}}) && (6.20)\\
\theta_3 &= acos(\frac{\ell_1^2 + \ell_2^2 - x_{pos}^2 - z_{pos}^2}{2 \cdot \ell_1 \cdot \ell_2}) && (6.21)
\end{aligned}$$

These three equations constitute the inverse kinematic equations of the robot. A small test of this model is described in section E, where it is concluded, that the found equations work as they are supposed to. It is however also noted that the MR suffers from mechanical uncertainties. It was also noticed, that the model generally suffered from the largest deviations when it was operating furthest away from its centre position.

With the verification of the inverse kinematic model the kinematic analysis of the MR has been concluded. It is now time to move to the subject of dynamic modelling, where the forces acting on the MR will be discussed.

Chapter 7

Dynamic leg model

In the following chapter the dynamic model of one of the legs of the MR will be considered. This model describes the movement of the leg when considering the forces and torques acting on it. Generally, the dynamic model of the legs can be considered to be composed of three sub models. The first model is the model of the servo motors, which takes in a reference angle as input, and the load torque from the robot in the joint as a disturbance, and outputs the angle, angular velocity and angular acceleration. The second model describes the load torque generated in each link, by means of the output from the servo motor model. Finally, the third model describes the forces acting on the end of the legs, when they strike the ground - the contact model. These three models will in the following be described more thoroughly. When these three sub models have been derived, a complete description of the dynamics of one of the legs has been obtained. Then a description of how the six leg models are connected to produce the complete system model follows.

7.1 Model of the servo motor

The first part of the model of the leg forces is the model of the servo motor. This model should be able to describe how the position, velocity and acceleration of the joint changes with time, when the load on the joint increases. That is, the servo motor model should be as described by figure 7.1

This load torque is generated when the masses of the legs are accelerated, or when the end effector strikes the ground. In 7.1.1, the assumptions and theory behind the proposed model are derived, and the proposed model is verified in section7.1.3.

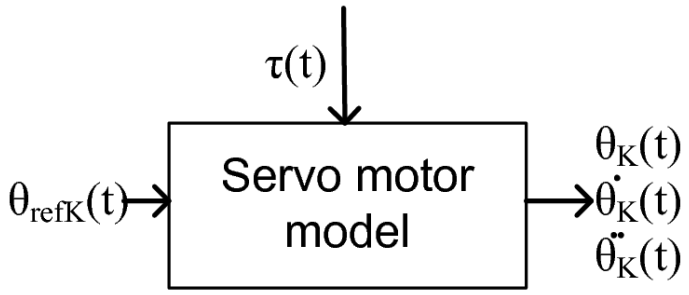

Figure 7.1: Shows the general schematic for the servomotor system model

7.1.1 Developing a servo motor model

In the following section, the servo motor model is derived. As seen from picture 7.1 three outputs are sought. These are the velocity, position and acceleration and can be derived in several ways. One approach is used in [22, p67], which provides good results regarding the position. However this approach directly differentiates the velocity to obtain the acceleration, and the velocity is assumed to be proportional to the angular displacement. When a new position reference is set, the velocity changes abruptly, which entails that the acceleration becomes huge (infinite if a continuous system is assumed). Instead another approach is used that is very similar to this approach.

The first step of the model is to make assumptions regarding the structure of the internal control loop of the servo motor. In [22] it is assumed that the velocity of the servo motors is proportional to the angular displacement. This assumption is modified so that the angular displacement forms a velocity reference instead:

$$\omega_{ref} = c \cdot (\theta_{ref} - \theta(k)) \tag{7.1}$$

This velocity reference is used to determine if power should be applied to the motor to minimize the angular displacement. When power is applied to the motor, it generates a torque, which again generates an angular acceleration around the motor axis. This entails that the angular acceleration should be proportional to the error in velocity.

$$\dot{\omega}(k+1) = b \cdot (\omega_{ref} - \omega(k)) \tag{7.2}$$

By combining 7.1 and 7.2, the following is obtained:

$$\dot{\omega}(k+1) = b \cdot (c \cdot (\theta_{ref} - \theta(k)) - \omega(k)) \tag{7.3}$$

$$\dot{\omega}(k+1) = b \cdot c \cdot \theta_{ref} - b \cdot c \cdot \theta(k) - b \cdot \omega(k) \tag{7.4}$$

Now that the acceleration is known, the velocity and position can be found by the following two formulas:

$$\omega(k+1) = \omega(k) + T_s \cdot \dot{\omega}(k) \tag{7.5}$$

$$\theta(k+1) = \theta(k) + T_s \cdot \omega(k) \tag{7.6}$$

$$\tag{7.7}$$

where T_s is the sampling time of the system. To better model the system, a set of constraints are included. The first constraints imposed on the system are the constraints on the maximum movement of the servo.

$$\theta_{Kmin} < \theta_K(k) < \theta_{Kmax} \tag{7.8}$$

Via measurements and datasheets, these are found to be $\theta_{Kmin} = -\pi/2$ and $\theta_{Kmax} = \pi/2$. The second constraint imposed on the system is the velocity limit of the servo motor.

$$\dot{\theta}_{min} < \dot{\theta}(k+1) < \dot{\theta}_{max} \tag{7.9}$$

Typically a no load max speed is given in the data sheet for a given servo. It is expected that this max speed is effected by the load torque.
To be able to discover if this max speed is affected by the load on the motor, a series of tests are made. The specifications can be found in section F. In this test, the servo motor was given an angular reference while loaded by several different torqueses. These tests are plotted on Figure 7.2 From this figure it is seen that the servo motor becomes slower the bigger the load

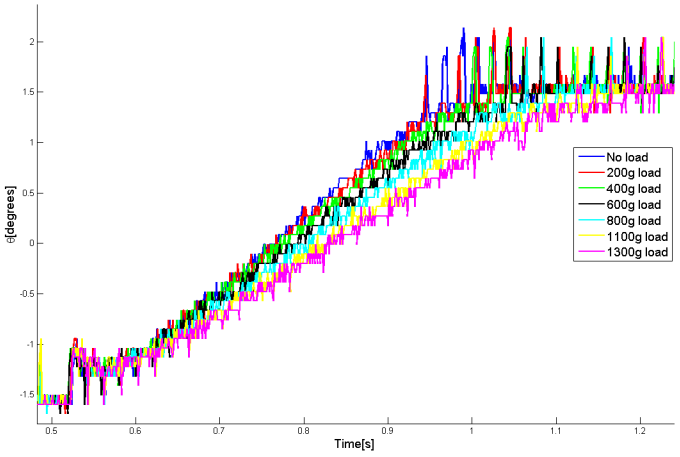

Figure 7.2: Shows the step responses when a load torque is added to the rotator of the servo motor

is. As the assumption is, that the load torque affects the maximum speed of the system, this speed is plotted against the load torque When these are plotted against each other Figure 7.3 is obtained: From this figure an approximately linear relationship can be seen. By fitting a

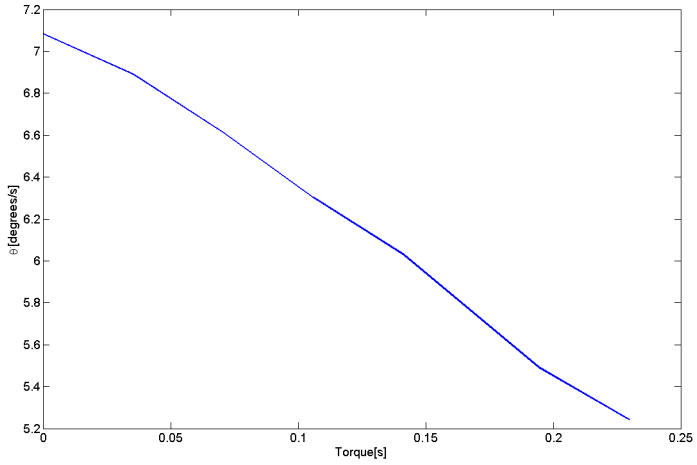

Figure 7.3: Shows the relationship between the load torques and maximum speeds of the system

first order polynomial to this data, the following line is obtained:

$$\omega(k)_{max} = 7.1574 - 8.2921\tau_l \tag{7.10}$$

where τ_l denotes the load torque

This entails, that a complete model of the three wanted parameters - acceleration, velocity and position - has been derived. The next step is to transform the equations from 7.4,7.5 and 7.6 into state space formulation.

7.1.2 State space formulation

The model is transformed into discrete state space form. The state space formulation is based on the three equations 7.4,7.5 and 7.6. By disregarding the constraints, a linear state space model can be found, based on three states:

$$\begin{bmatrix} \theta(k+1) \\ \omega(k+1) \\ \dot{\omega}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & T \\ -b \cdot c & -b & 0 \end{bmatrix} \cdot \begin{bmatrix} \theta(k) \\ \omega(k) \\ \dot{\omega}(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ b \cdot c \end{bmatrix} \theta_{ref}(k) \tag{7.11}$$

As the interesting variables - the position, velocity and acceleration - are all states of the system, the output equation is easily derived:

$$\begin{bmatrix} \theta(k) \\ \omega(k) \\ \dot{\omega}(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \theta(k) \\ \omega(k) \\ \dot{\omega}(k) \end{bmatrix} \tag{7.12}$$

With these last equations all set, a complete servo model has been proposed. A realization of this model should be derived in simulink, and is composed by means of a normal state

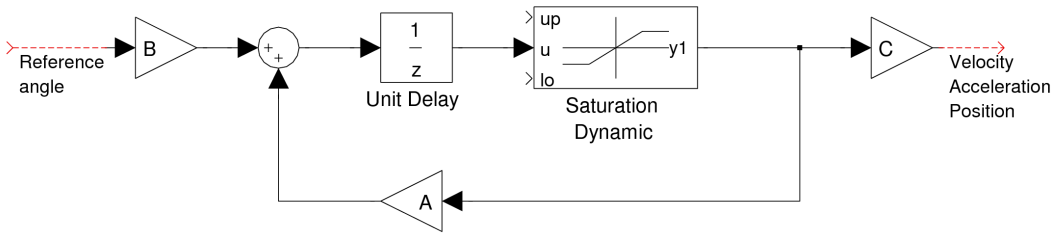

Figure 7.4: Shows the simulink model of the constructed simulation model for the servo motor

space formulation, with constraints affecting the system states. It should be noted, that the equations described in 7.11 and 7.12 are the equations governing the dynamics of one servo motor, where the parameters b and c describe the individual characteristics of this certain servo. The next step is to derive these parameters. As all 18 servo motors are identical, it is in the following assumed that the results obtained can be generalized to all the servo motors.

7.1.3 Obtaining parameters and verifying the servo model

In the following section, the derived servo motor model is verified using measurements. To be able to verify if the model works, parameter estimation should be conducted. By using the sum of squared errors, and the simplex method via the "fminsearch" function in MATLAB, the following parameters for the system are obtained:

$$b = 216.6769 \tag{7.13}$$

$$c = 36.6290 \tag{7.14}$$

When these parameters are used in the derived model and plotted against the steps from figure 7.2 the following is obtained: The first thing that is noticed is that the model is unable to describe the jump in reference at t=0.5s where the position reference is changed. It is however suspected that this abrupt change in position is actually an anomaly produced from the potentiometer, and not an actual physical displacement. However it is seen from the figure, that a good approximation, of the dynamics of the steps, has been achieved. It is also seen that this model works as the servo becomes loaded, which again supports the assumption, that the way a load torque affects the output is by limiting the maximum angular velocity.

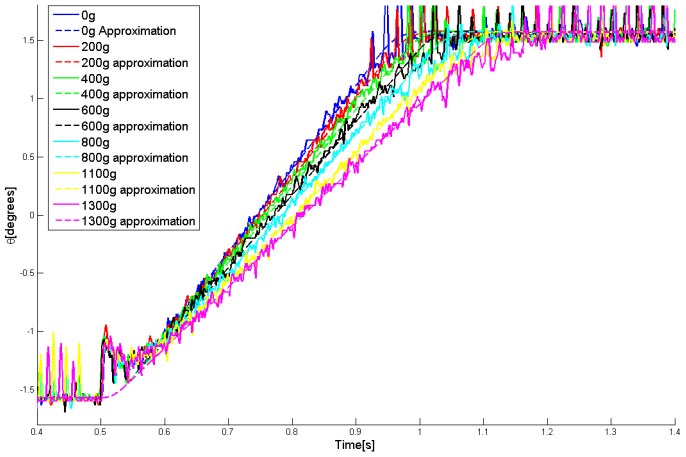

Figure 7.5: Shows the steps shown on Figure 7.2 but with added approximations from the derived model

7.2 Dynamics of the mechanical platform

Now that a model of the servomotors has been derived, it is time to expand the limited kinematic viewpoint used in 5 to include forces, velocities and accelerations. This model describes the dynamics governing one of the legs of the MR. To describe the behaviour of the entire MR, a total of six identical models is made, one for each leg. These six models are combined in section 8. The dynamic models of the legs provide one important piece of information - the unknown load torque described in 7.1. This model should be able to predict the forces acting on the robot, when the servomotors of the legs are moved. The derivation is based on a method described in [14, p165-185] as the iterative Newton-Euler dynamics algorithm. It takes the velocities, accelerations and positions of the servo motors, which are known from section 7.1, and describes the torques acting on each joint to produce these accelerations. It consists mainly of two steps, the forward iterations, and the backward iterations, which are presented shortly below.

7.2.1 Outward iterations

The method is based on making two sets of calculations for each joint divided into two "runs". In the first run the rotational and linear velocity, and the linear and rotational acceleration of the centre of mass is found by iteratively starting with link 1 and moving out to link n. By knowing the accelerations and velocities it is possible to calculate the inertial forces acting on each centre of mass, by using Newton's and Euler's equations. This entails that a total of 6 equations are needed for the outward iteration. These equations can be seen below, where the final two are directly derived from Newton's and Euler's equations. They can also be

found in [14, p176].

$$
\begin{aligned}
{}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R \cdot {}^{i}\omega_i + \dot{\theta}_{i+1} \cdot {}^{i+1}\hat{Z}_{i+1} && (7.15)\\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}R \cdot {}^{i}\dot{\omega}_i + {}^{i+1}_{i}R \cdot {}^{i}\omega_i \times \dot{\theta}_{i+1} \cdot {}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1} \cdot {}^{i+1}\hat{Z}_{i+1} && (7.16)\\
{}^{i+1}\dot{v}_{i+1} &= {}^{i+1}_{i}R \left({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1} \right) + {}^{i}\dot{v}_i \right) && (7.17)\\
{}^{i+1}\dot{v_{C}}_{i+1} &= {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{Ci+1} + {}^{i+1}\omega_{i+1} \times \left({}^{i+1}\omega_{i+1} \times {}^{i+1}P_{Ci+1} \right) + {}^{i+1}\dot{v}_{i+1} && (7.18)\\
{}^{i+1}F_{i+1} &= m_{i+1} \cdot {}^{i+1}\dot{v_{C}}_{i+1} && (7.19)\\
{}^{i+1}N_{i+1} &= {}^{C_{i+1}}I_{i+1} \cdot {}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\dot{\omega}_{i+1} \times {}^{C_{i+1}}I_{i+1} \cdot {}^{i+1}\omega_{i+1} && (7.20)
\end{aligned}
$$

In these equations quite a number of variables and parameters is included. These can be described without indexes:
"i" indicates the joint number.
θ is the angle of the joint
$\dot{\theta}$ is the angular velocity of the joint
$\ddot{\theta}$ is the angular acceleration of the joint
R is the rotational matrix
ω Is the angular velocity.
$\dot{\omega}$ is the angular acceleration
$\dot{v}$ is the velocity of the link
$\dot{v_c}$ is the velocity of the centre of mass of the link
F is the force acting on the centre of mass
N Is the torque acting around the centre of mass
m is the mass of the link
I is the inertia tensor of the link
$\hat{Z}$ is a vector pointing in the Z direction in the link
P_C Is the vector pointing to the centre of mass
P is the vector pointing to the next link.

These 6 equations constitute the forward iteration of the Newton-Euler iterative dynamics algorithm. That is, by means of equations 7.15-7.20 it is possible to calculate all angular accelerations, speeds and inertial forces by moving from link to link starting from link 1. To be able to compute the forces acting on each joint, the inward iteration is applied

7.2.2 Inward iterations

Now that all accelerations, speeds and inertial forces are known for the centre of mass, the resulting forces applied to the joints are calculated. This is done in a similar recursive way.

By moving from the link farthest away and inward, at each link applying equation 7.21-7.23. These equations can also be found from [14, p176]

$$
\begin{aligned}
{}^{i}f_i &= {}^{i}_{i+1}R \cdot^{i+1} f_{i+1} +^{i} F_i && (7.21)\\
{}^{i}n_i &= {}^{i}N_i +^{i}_{i+1} R \cdot^{i+1} n_{i+1} +^{i} P_{Ci} \times^{i} F_i +^{i} P_{i+1} \times^{i}_{i+1} R \cdot^{i+1} f_{i+1} && (7.22)\\
\tau_i &= {}^{i}n_i^T \cdot^{i} \hat{Z}_i && (7.23)
\end{aligned}
$$

This entails, that it is finally possible to derive a vector of torques and forces, acting on each joint of the robot legs. In the inward iteration, the parameters and variables used are mostly the same as in the outward iteration. The ones which are not in common are: f is the force acting on the joint.
n is the torque acting on the joint.
τ is the torque acting around the axis of rotation of the joint.

It is the torque τ which can be seen as the moment loading the servo motor, and it is thus this parameter that should be fed back to the servo motor loads described in section 7.1.
From the set of equations defining the inward iteration, it is seen that, when calculating n_i and f_i, it is necessary to know a force f_{i+1}. When looking at the link in which the end effector is placed, this force does not come from another link, and is thus external. This force is the contact force on the end effector when the robot hits the ground. That is, when the leg is not in contact with a surface, the force is zero, and when it is in contact, the force is nonzero. This entails that a derivation of the magnitude of this force should be made. This is referred to as the contact model, and the derivation is conducted in 7.3. Another thing that is noticed from the equations is the presence of several parameters describing the dynamics of the links. These are: the position of the centre of mass, the inertia tensor and the mass of the links. These parameters are found from Appendix B, where a 3D model of the robot is developed in Solid works.

7.2.3 Iterative versus closed form

When an iterative method as the ones given by equations 7.15 to 7.23 is used, it is possible to use it in two forms. One which is iterative and one which is in closed form. As the derivation of these closed form equations are quite cumbersome, the iterative approach is used. This entails, that the model is implemented in simulink, as a series of interconnected iterative blocks. Three are needed in the forward direction, and three in the backward direction.

In these blocks, the equations from the Outward and Inward iterations are placed. If the first link is considered, the following code can be found inside the forward iteration:

```
function [w1,wm1,v1,vm1,F1,N1,Vcm1] =fcn(theta,thetam,thetamm,w0,wm0,v0,v
```

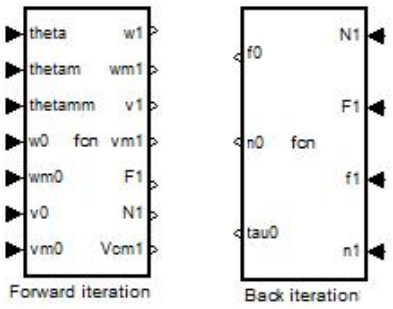

Figure 7.6: Shows the general structure of the iterative blocks of the dynamics equation

```
Ic=Leg1CoxaIc;
Pc=Leg1CoxaPc;
P=Leg1CoxaP;
m1=Leg1Coxam1;
Z=Leg1CoxaZ;
alp=0;
R=[cos(theta)           -sin(theta)          0 ;
   sin(theta)*cos(alp) cos(theta)*cos(alp) -sin(alp);
   sin(theta)*sin(alp) cos(theta)*sin(alp) cos(alp)]^-1;
w1=R*w0+thetam*Z;
wm1=R*wm0+R*cross((w0),(thetam*Z))+thetamm*Z;
v1=R*(v0)+cross(w1,P);
vm1=R*(cross(wm0,P)+cross(w0,cross(w0,P))+vm0);

Vcm1=cross(wm1,Pc(1:3))+cross(w1,cross(w1,Pc(1:3)))+vm1;

F1=m1*Vcm1;
N1=Ic*wm1+cross(w1,(Ic*w1));
end
```

In block describing the inward iteration, the following code can be found.

```
function [f0,n0,tau0] = fcn( theta, N1,F1,f1, n1,Leg1CoxaIc,Leg1CoxaPc,
Ic=Leg1CoxaIc;
Pc=Leg1CoxaPc;
P=Leg1CoxaP;
m1=Leg1Coxam1;
Z=Leg1CoxaZ;
```

```
alp=pi/2;

R=[cos(theta)           -sin(theta)          0 ;
   sin(theta)*cos(alp) cos(theta)*cos(alp) -sin(alp);
   sin(theta)*sin(alp) cos(theta)*sin(alp) cos(alp)];

f0=R*f1+F1;
n0=N1+R*n1+cross(Pc,F1)+cross(P,R*f1);
tau0=n0'*Z;
```

It is seen, that the rotational matrix is not the same. This is due to the fact, that the outward iterations are conducted from frame 0 to the frame of the last joint. And the inward from the frame of the end effector, to the frame of the first joint.

Now that the dynamic model, describing the forces acting on the legs of the robot, has been derived, it is time to look at the external force affecting the legs of the MR. This is done in the following section concerning the contact model.

7.3 Contact model

The final part of the modelling, which has to take place, is the derivation of the force, acting on the end effectors from the ground, when the robot touches the supporting surface. There are several ways to model this behaviour, and can be a very complex subject to cover. One should in general treat the movement of the end effector as being able to move about in space as defined by Newtonian mechanics, but with applied constraints. When a constraint is broken, impact occurs and the basis for the Newtonian mechanics changes. The force acting on the end effector f_{end} changes as a function of in which phase the leg is. If it is in mid air, f_{end} equals zero, as no forces act on the end effector. If it is in resting contact, the force must be just big enough to make the acceleration towards the object in consideration equal to zero (that is; no interpenetration allowed), and if it is in a phase of collision, the impact force from the leg must spawn an opposite reaction from the ground. To simplify the modelling, an assumption is introduced. It is assumed that collision only occurs between the end effector and the ground. This entails that the robot is treated as 6 points in space, instead of a complete body.

There are several ways to describe such a contact model. Among others one found in [27, p40-p62] where an example and implementation can be seen in [28]. In this approach, the

case of collision and resting contact and collision are treated differently. The case of resting contact is based on deriving the contact forces by means of a fast numerical algorithm. In the case of collision, the model deviates from a force analysis, and emulates the ground reaction force as a momentum impulse [27, p42]. This is due to the fact that a force can only change acceleration, and thus needs some time to act to change the velocity of the colliding objects. As the derived dynamic model is explicitly in a force torque domain, another approach is needed. The approach that is chosen is based on an assumption stating that there between the ground and contact point exist a spring damper system. One might interpret this as if there are springs attached to the legs of the robot, or if the floor surface is somewhat flexible. When the robot hits the ground, the spring damper system also comes in contact with the ground. This results in a force which moves the end effector towards the surface. To make the spring yield a force, a small amount of interpenetration ϵ needs to be allowed. Another effect that needs to be described is the friction between the surface and the end effector. If the end effector strikes the ground in an angle, it will not slide much on the ground, but come to a halt quickly. To simplify this behaviour, it is assumed that no sliding occurs. It is decided to model this friction as viscous friction, as it is simple to include in the model, and by providing a sufficiently big friction coefficient, the end effector will come to a halt fast. To model this behaviour, the end effector is modelled as being mounted by two additional dampers. A concept of how the system is visualised can be seen from Figure 7.7 To be able to implement such a spring damper system, the spring and damper coefficients k_1,c_1,c_2 and c_3, should be found. The first coefficients that are to be considered are the ones of the spring damper system which is to prevent interpenetration, that is, k and c_1. As it is unavoidable that a small amount of interpenetration occurs when the spring damper model is used, a constraint for this interpenetration should be determined ϵ_{max}. To ease the derivation of the parameters, it has been chosen to derive this constraint while the system is in stationary state. If the end effector link is considered, one might interpret this interpenetration as a shortening of the end effector. The change in the length of the end effector should be considerably smaller than the length of the end effector l_e.

$$\epsilon_{max} << l_e \tag{7.24}$$

From the 3D model developed in B it can be seen that the end effector has a total length of 14.35 cm. This entails that ϵ_{max} should be at least 100 times smaller or

$$\epsilon_{max} < 14.35/100cm = 1.435mm \approx 1mm \tag{7.25}$$

If the endpoint is assumed to be a single mass affected by the spring and damper forces, a second order system is obtained:

$$m \cdot \ddot{\epsilon} = -c_1 \cdot \dot{\epsilon} - k \cdot \epsilon + F_{ext} \tag{7.26}$$

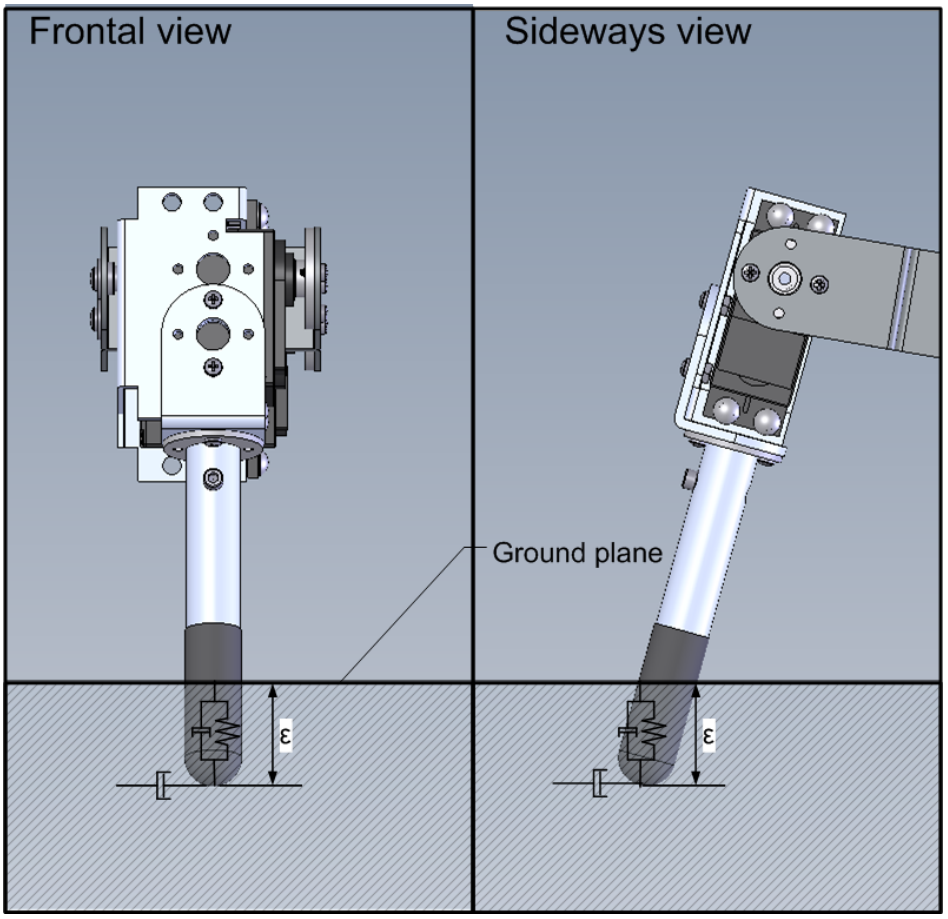

Figure 7.7: Shows a visualization of how the impact model can be modelled using springs and dampers

where:
m is the mass
c_1 is the damper coefficient
k is the spring coefficient
F_{ext} is an external force acting on the mass

This spring damper system describes how the penetration distance ϵ changes as a function of the applied force F_{ext}. When the system is in rest, the following equilibrium applies:

$$k \cdot \epsilon = F_{ext} \tag{7.27}$$

In a worst case scenario the constant k should be sufficiently big, so that $\epsilon < \epsilon_{max}$. To find the required constant, a worst case scenario for the external force should be determined. As the derivation is based on a steady state assumption, this worst case should be considered in this context. As described above, contact wise, the MR can be thought of as six points of potential contact. The worst case situation is when the MR is standing in such a way that all weight is put upon a single leg. In such a situation, the external force from 7.27 to:

$$F_{ext} = m_{MR} \cdot g \tag{7.28}$$

where: m_{MR} is the total weight of the MR=2.437kg
g is the gravitational acceleration 9.81

When these values are inserted, the external force can be calculated.

$$F_{ext} = 2.437 \cdot 9.81 = 23.9N \tag{7.29}$$

If this is inserted in 7.27 the spring constant can be derived.

$$k \cdot 0.1cm = 23.9N \tag{7.30}$$

$$k = 239N/cm \tag{7.31}$$

$$\tag{7.32}$$

Now that the spring constant has been derived, it is time to find the damper coefficient. This coefficient should describe how much the end effector jumps up and down. For a second order system, this effect is described by the coefficient ζ, which for a mass spring damper system can be formulated as:

$$\zeta = \frac{c}{2 \cdot \sqrt{m \cdot k}} \tag{7.33}$$

The ζ coefficient indicates if a system is over damped, critically damped or under damped. A critically damped system has ζ=1, and produces the fastest system without oscillatory behaviour. As oscillations are not wanted, it has been chosen to use a critically damped system. This entails that ζ=1 and thus the damper coefficient can be found from 7.34

$$\zeta = \frac{c}{2 \cdot \sqrt{m \cdot k}} 1 = \frac{c}{2 \cdot \sqrt{2.437kg \cdot 239N/cm}} \tag{7.34}$$

$$c = 2 \cdot \sqrt{2.437kg \cdot 239N/cm} \tag{7.35}$$

$$c = 2 \cdot \sqrt{2.437kg \cdot 239 \frac{\frac{kg \cdot m}{s^2}}{cm}} \tag{7.36}$$

$$c = 2 \cdot \sqrt{2.437kg \cdot 239 \frac{\frac{kg \cdot m}{s^2}}{0.01m}} = 241.33kg/s \tag{7.37}$$

$$\tag{7.38}$$

This entails that the two constants have been found, and the next parameters that should be derived are the damping coefficients modelling the friction in the sideways direction. The first observation that should be made is that the friction should be equal in the frontal and sideways direction, that is:$c_2 = c_3$. When determining the constants, the first thing that needs to be determined is what type of constraint is needed. The constraint can either be in the form of a max distance from the place the end effector reaches the ground, or a max time for the end effector to settle. As a constraint of 1mm is already made in the vertical direction the

same constraint is made in the sideways direction. The magnitude of these constants can be estimated by making a worst case assessment. The robotic leg is assumed to be a point mass, with the weight of the entire robotic leg, and travelling with a maximum velocity v_{max}. This point mass hits the ground, and needs to be stopped within 1mm of the impact area. The first thing that should be derived is the maximum velocity. The only actuator which can affect the sideways motion of the limb is the Coxa servo. This entails that the maximum velocity is derived as the total length of the robotic leg combined with the maximal angular velocity of the Coxa servo motor:

$$v_{max} = \omega_{max\ coxa} \cdot l_{leg} = 7.1574[rad/s] \cdot (14.35 + 5.6 + 3.8)[cm/rad] = 170cm/s \quad (7.39)$$

The combined weight of the robotic leg is found from the derived 3D model described in B:

$$m_{leg} = 149.20g + 29.75g + 91.52g = 270.47g = 0.270kg \quad (7.40)$$

Now that these parameters are found, the damper constant should be derived. When considering the point mass, the only sideways forces acting on it is the friction. If x denotes the displacement in any of the sideways directions, and the subscript d denotes the displacement, the following equation can be found:

$$m \cdot \ddot{x_d} = c \cdot \dot{(x_d)} \quad (7.41)$$

$$\ddot{x_d} = \frac{c}{m} \cdot \dot{(x_d)} \quad (7.42)$$

$$(7.43)$$

That is, the acceleration is determined by the speed of the displacement By integration, this yields:

$$\dot{x_d} = \int_0^t \frac{c}{m} \cdot \dot{(x_d)} dt \quad (7.44)$$

$$(7.45)$$

When the point mass has come to a halt, it is known that the integrated velocity should equal the velocity with which the point mass strikes the ground plane.

$$v_{max} - \int_0^t \frac{c}{m} \cdot \dot{(x_d)} dt = 0 \quad (7.46)$$

$$v_{max} - \frac{c}{m} \int_0^t \cdot \dot{(x_d)} dt = 0 \quad (7.47)$$

$$v_{max} - \frac{c}{m} x_d = 0 \quad (7.48)$$

$$(7.49)$$

By inserting the maximum allowed displacement x_d as 1 mm and the maximum velocity and the mass of the robotic leg, an assessment of the damper coefficient can be found.

$$c = \frac{v_{max} \cdot m}{x_d} = \frac{170cm/s \cdot 0.270kg}{0.1cm} = 459kg/s \quad (7.50)$$

This entails that all parameters of the contact model have been found, and an implementation in MATLAB should be made. As the system can be said to consist of being in several phases, it is natural to model it as a state machine in MATLAB. When in the first phase the force acting on the end effector is zero, and when, in contact with the ground, the friction forces described by 7.26 and 7.41 apply. The implementation of these phases can be seen from Figure 7.8

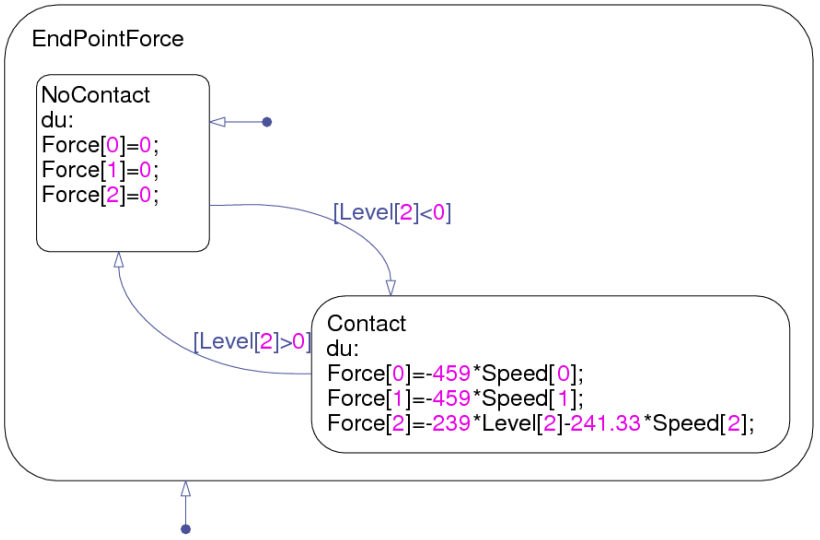

Figure 7.8: Shows the state model of the endpoint forces implemented as a MATLAB state flow chart

To verify the above calculations, a simulation is conducted with a point mass. In the first simulation, a point mass of 2.4037 kg is dropped from 2 cm on a ground plane. This mass should obtain an interpenetration of maximum 1mm when in steady state. The results of this simulation can be seen from figure 7.9 The next simulation is conducted in such a way that the end effector is situated 1mm below ground, with a start x velocity of 170 cm/s. In this situation, the end effector needs to stop within 1mm of the start position. The results of this simulation can be seen from figure.

From these simulations, it can be seen that the impact model is able to stop the interpenetration of the ground plane, with a steady state penetration of less than 1mm. It can also be seen that the friction model stops the end effector before it reaches 1mm. In this last simulation, one should notice the small time interval in which the end effector is stopped. This small interval is due to the fact that only a small amount of friction is allowed, and the start velocity is very big. If the point mass, travelling with a velocity of 170cm/s needs to be stopped within 1mm, it needs to be slowed down very quickly. These tests indicate that the contact model works as intended.
With the completion of the contact model, the set of sub model defining the entire dynamic system model is now complete. In the following chapter, these sub models are combined into

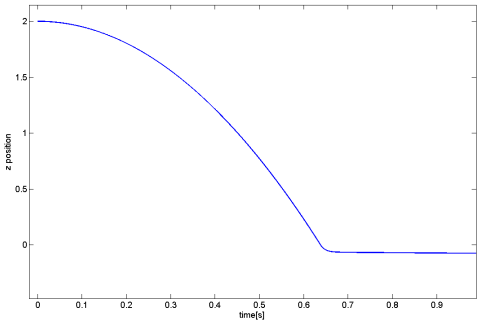

Figure 7.9: Shows a simulation of the impact model, when a point mass is dropped on a ground plane from a height of 2cm

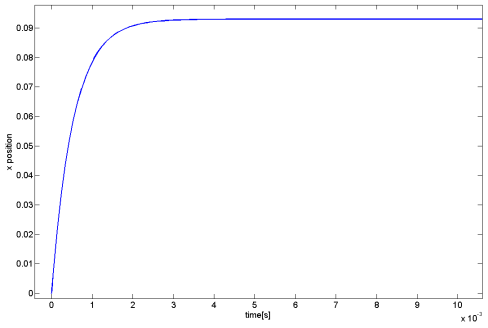

Figure 7.10: Shows a simulation of the impact model, when a point mass is situated below ground plane with a start velocity of 170cm/s

one large system model.

Chapter 8

Combining models

By having defined all necessary models, it is vital to describe how these are interconnected to create the complete system model. As already stated in II the model is generally composed of several sub models, which by themselves only describe parts of the behaviour of the hexapod robot. Generally speaking, the MR is composed of a body, with 6 legs attached. The only forces acting directly on the body, are the forces acting at the points where the legs are attached, and the gravitational force acting on the centre of mass. This situation can be seen from Figure 8.1.

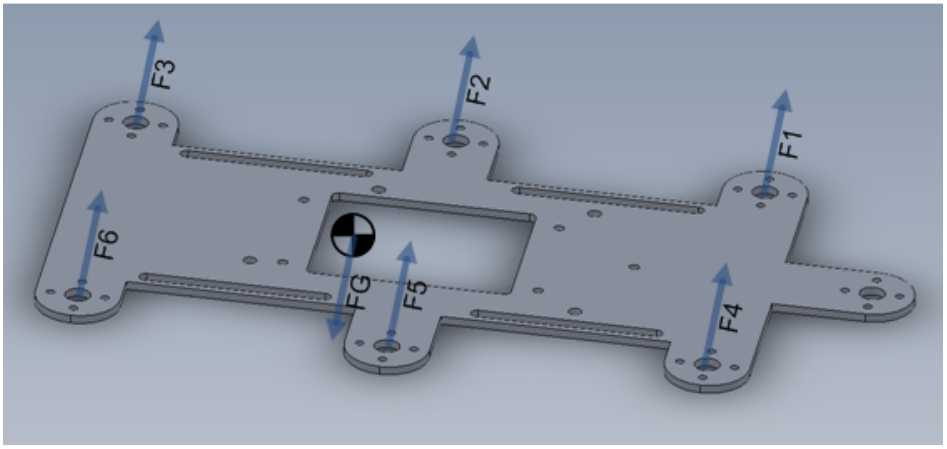

Figure 8.1: Shows the forces acting on the robot body, when considering it as a free body(Notice that the forces F1-F6, can point in any direction, depending on the leg configuration)

These six forces are all caused by the movement of the legs of the MR, and can be predicted by the model described in section 7.2. To make a description of how the entire system behaves, it is chosen to interpret the system as a rigid body, which is affected by 6 forces coming from the legs, plus a force coming from the force of gravity.

8.1 Robot as a rigid body

Generally the MR is treated as a being a rigid free body moving in a 3d coordinate system, as described in section 7. This free body can be effected by 6 forces plus the force of gravity. The motion of such a free body can generally be described by its position and orientation, and the derivates of these two quantities, such as velocity and acceleration. As with all free bodies, the linear acceleration can be described with newtons equation.

$$\dot{v} = \frac{1}{m_{MR}} \sum F_i \tag{8.1}$$

where:
$\dot{v}$ denotes the acceleration vector of the centre of mass of the MR.
m_{MR} denotes the mass of the mobile robot

There exists a rotational analogy to newtons equation, describing the angular motion of rigid bodies, called Eulers equation. This equation can be seen from 8.2

$$\tau =^C I \cdot \dot{\omega} + \omega \times^C I\omega \tag{8.2}$$

where:
${}^C I$ is the inertia tensor of the rigid body.
τ is the total torque acting around the centre of mass of the body

This torque is caused by the forces from the legs. When the forces are unevenly distributed, a rotation will occur around the centre of mass. This torque can also be described by the following equation:

$$\tau = \sum \tau_i = \sum (r_i - x(t)) \times F_i \tag{8.3}$$

Where:
τ is the total torque acting around the centre of mass of the robot.
τ_i is the torque acting resulting from a force acting on one of the six endpoints.
r_i is the coordinate vector containing the place where the force acts
x(t) is the coordinate vector containing the location of the centre of mass

By using equation 8.1, 8.2 and 8.3 it is possible to calculate the angular and linear accelerations of the centre of mass of the MR. By combining this information with the kinematics of the MR a description of how the MR moves can be obtained.

Now that a system model have been obtained, it is time to introduce the concept of stability for legged robotics. This is done in the following section, by introducing the Zero Moment Point(ZMP).

Chapter 9

Stability of the Mobile Robot

Now that a dynamic system model has been devised in the previous chapters, it is time to make a description of how stability is represented, as this is a vital piece of information, when controlling the mobile robot. The definition of such a concept is not easily described by the normal terms used within the field of control theory. Normally system stability would be described by means of the pole positions of transfer functions. These parameters are however only defined for linear systems. If a linearization of the system were conducted, it would rely on an assumption of a linear relationship between servo angles, and robot position, which would be wrong. This entails that another definition is required. In [2, p29] a distinction between statical and dynamic stability is made.

9.0.1 Static stability

A definition of static stability can be found from [2, p29] and states that: *If a limbed robot is designed so that its balance is maintained at all times even if all its legs were to freeze in position, then the robot is said to exhibit static stability*. To obtain static stability it is necessary for the centre of gravity of the robot to stay within the "Convex hull", of the support polygon defined by the legs currently in touch with the ground. The convex hull can be seen as a region in space constructed as the smallest convex region containing all contact points points An illustration of the convex hull, can be seen from 9.1

9.0.2 Dynamic stability

From [2, p30] a note of dynamic stability describes it as: "if the centre of gravity of the robot is allowed to move outside of the convex hull of the support polygon and the robot moves in a

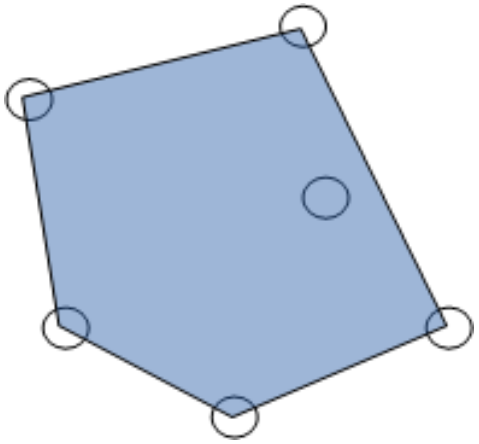

Figure 9.1: Shows the convex hull of the hexapod robot, when all six legs are in contact with the ground

controlled manner, the robot is said to exhibit dynamic stability". This definition is definately valid, but it does not provide a way of detecting for dynamic stability. To explore dynamic stability of the robot, it is necessary to derive another measure, which is the Zero Moment Point(ZMP). The zero moment point is a quantity used to describe if the robot is in balance. A good explanation of the concept can be found from [29, p18-36]. The zero moment point is related to the ground reaction force. In 7.3 it is described that the ground reaction force works perpendicular to the ground plane(If friction is neglected). This entails that any x or y moment working about the centre of gravity of the robot can be annulled by applying the contact force at a certain point, If this point is located inside the convex hull of the robot it is called the ZMP and if outside it is called the Fictious ZMP(FZMP). If a FZMP is present, it would require the ground contact force to be applied outside the POS to compensate for x and y moments. As the ground reaction force can only act within POS, a moment around x and y will result, which again entails a rotation of the system/loss of balance. A more formal interpretation have been found in [30, p164](Originally posed by [31]), which is a summational paper of the ZMP written by Miomir Vukobratovi.

p is the point that T_x=0 and T_y=0, T_x,T_y represent the moments around the x- and y-axis generated by reaction force F_r and reaction torque T_r, respectively. The point p is defined as the Zero Moment Point(ZMP). When ZMP exists within the domain of the support surface, the contact between the ground and the support leg is stable.

$$p_{ZMP} = (x_{ZMP}, y_{ZMP}, 0) \in S \tag{9.1}$$

Where P_{ZMP} denotes a position of ZMP. S denotes a domain of the support surface. This condition indicates that no rotation around the edges of the foot occurs

A way of calculating the position of x_{ZMP} and y_{ZMP} is fond from [29, 35], which states:

$$x_{ZMP} = \frac{\sum_i^n \left(m_i \left(x_i \left(\ddot{z}_i + g_z\right) - z_i \left(\ddot{x}_i + g_x\right)\right) - I_{iy} \cdot \omega_{iy}\right)}{\sum_i^n \left(m_i \left(\ddot{z}_i + g_z\right)\right)} \tag{9.2}$$

$$y_{ZMP} = \frac{\sum_i^n \left(m_i \left(y_i \left(\ddot{z}_i + g_z\right) - z_i \left(\ddot{y}_i + g_y\right)\right) - I_{ix} \cdot \omega_{ix}\right)}{\sum_i^n \left(m_i \left(\ddot{z}_i + g_z\right)\right)} \tag{9.3}$$

In this study, a simplification of the calculation of the ZMP, used in [22], will be utilized. This simplification assumes that the masses of the links are uniformly distributed about the centre of masses.

$$x_{ZMP} = \frac{\sum_i^n \left(m_i \left(x_i \left(\ddot{z}_i + g_z\right) - z_i \left(\ddot{x}_i + g_x\right)\right)\right)}{\sum_i^n \left(m_i \left(\ddot{z}_i + g_z\right)\right)} \tag{9.4}$$

$$y_{ZMP} = \frac{\sum_i^n \left(m_i \left(y_i \left(\ddot{z}_i + g_z\right) - z_i \left(\ddot{y}_i + g_y\right)\right)\right)}{\sum_i^n \left(m_i \left(\ddot{z}_i + g_z\right)\right)} \tag{9.5}$$

That is, the balance/stability of the system depends on the position of the ZMP. If the ZMP is located within the POS, the hexapod won't experience a rotation which eventually will make it turn over. This entails that the developed control strategy should always entail that the ZMP never leaves the POS in a given gait cycle. This entails that the distance from the ZMP to the edge of the POS can be used as a measure of stability. This is utilised when designing the controller for the system in part IV.

Now that a complete system model has been derived, it is time to verify that the model describes the physical system. This will be done in the following section.

Chapter 10

Model verification

The following section concerns the verification of the simulation model. Normally such a verification process would use a data set as a reference, and see if the model is able to predict the outcome, within a certain tolerance. This is however a problem, as the MR is not equipped with any sensors, capable of determining if the forces, torques or accelerations are correct. A small test were conducted to see if a verification of the dynamic model of one of the legs of the MR could be verified by means of measureing the current draw of the legs. This is described in section F, but unfortunately no valid results could be made. This entails, that a real model verification process cannot be carried out. In stead a verification of the model is carried out on a basis of simulations, and descriptions of the expected results. It is clear that this type of verification is not optimal, but as many of the concepts of simple mechanics come quite easily to human beings. This verification can be seen in section G, where it is concluded that the model is able to describe the "behavour of a sixlegged walking machine". This "six legged walking machine" were however modelled by the parameters obtained from the 3D model of section B. In this section, it was however also be discussed that the model would benefit from another friction model and that the chosen contact model based on a spring damper system slows down the simulation of the system quite severely.

Part conclusion

In the previous part of the report, the concept of making a mathematical model of the robot were considered. The model consisted of three submodels. The kinematic model, the inverse kinematic model, and the dynamic model. The two kinematic models were verified, while no clear verification could be obtained from the dynamic model. This was mainly due to the fact that the mobile robot were not equipped with any sensors, from which definitive conclusions could be reached. This entailed that the dynamic model were verified as a "six legged walking machine" from a set of simulations.

Part III

Software and intelligence

Chapter 11

Software and intelligence

In the following chapter, the intelligence and software structure of the MR is considered. As described in section 3, the programming of the MR should be able to use the sensor information provided from the camera to make various decisions, and convert this information to signals controlling the motors. Besides simple linetracking capabilities, it is this system that should make all decisions regarding where to go, if a crossroad is present, how to act, if the line dissapears, and even finding the best way through the track. In the following chapters, a description of the designed software system is conducted, then a description of how the MR should find it's way around the track is made, and finally a description of the implemented vision algorithm can be seen.

11.1 Preliminary analysis of software system

In this chapter the structure of the designed software system of the MR is considered. This system should be able to make the MR move about on the obstacle course, handle problems like the loss of a line reference. Keeping track of the position of the MR and various other problems. Before a system is presented, a preliminary analysis will be made.

11.2 Definition of terms

In [1, p181] the problem of how the MR should act, based on the sensor inputs, is described as the problem of navigation, which can be specified as how the MR should act to achieve

it's goals. The act of navigation can generally be decomposed into four subproblems as seen from figure 11.1.

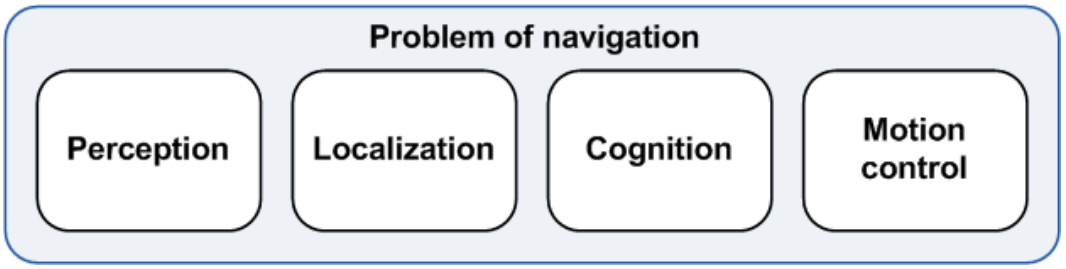

Figure 11.1: Shows the four problems of navigation

These four subproblems are the ones of Perception, Localization, Cognition and Motion control. These subproblems can be described by the following:

1. Perception
   The robot must interpret its sensors to extract meaningful data. For example indicate where the black line is by means of the camera

2. Localization
   The robot must determine its position in the environment

3. Cognition
   The robot must decide how to achieve its goals

4. Motion control
   The robot must modulate its power outputs to achieve the desired trajectory

It must be understood, that the the latter two should be interpreted to not only include the small term goal of staying on the line, but also the long term goal of moving from one place on the track to another, and even the final goal of moving from the start position, to the end. This entails that in the following chapters, a description of the Perception, Localization and cognition aspects of the developed software system is made. The last of these problems, the problem of motion control will be described in part IV, which concerns the short term goal of staying on the black line.

Chapter 12

Perception - Computer vision

The first part of the problem of navigating on the obstacle course is perception as seen from Figure 12.1 In section 11.1 it was described that the problem of Perception could be seen as the way the MR interprets its sensor information to derive meaningful features. The problem of perception of this thesis is how the MR should interpret the vast amounts of information provided by the CMOS light sensitive sensor of the camera.

The vision system should be able to provide the mobile robot with some of the capabilities that the human eye provides for us humans. The computer vision system should be able to provide the MR with a set of references that describe the black line of the obstacle course. For a human being, the visual task of finding the black line of the obstacle course might seem simple, but when such a task has to be implemented on a microprocessor, one discovers that a quantification of the steps involved in solving such a problem, quickly becomes complex. The main focus of the algorithm should be to make the robot able to generate a trajectory that should be followed so that the line is tracked. As the assignments of the obstacle course are many and varied, several other features of the vision system are needed, and generally the following should be considered:

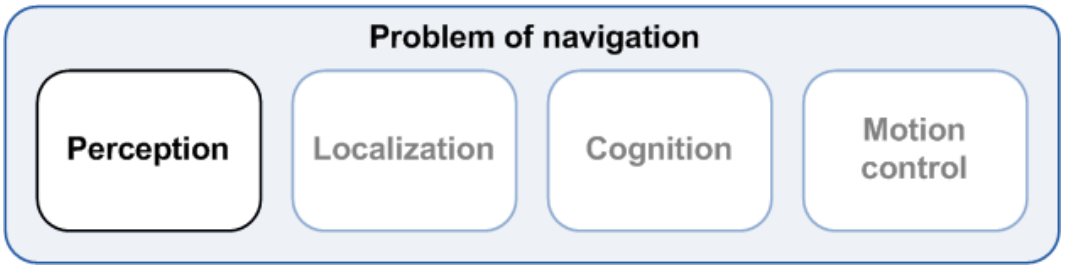

Figure 12.1: Shows the four sub problems of Navigation. The segment which is analysed in the following chapter is the first one, the problem of perception

- Line tracking
  As already described, the vision system should be able to provide the MR with a trajectory that should be followed for optimal line tracking.

- Crossroad detection
  Another required feature is crossroad detection. It should be possible to detect if a crossroad is present. This feature should however be implemented on the on board computer, as the computer vision system should not make "Judgements", but only function as an intelligent sensor.

- Expandable
  It is necessary that the developed vision system is expandable, so that new features are easily integrated. Several of the tasks on the obstacle course could be simplified by means vision capabilities, and the developed algorithm should be easily modified to provide the required capabilities, e.g. ball recognition or discerning a blue flower from a red flower.

- Robustness
  As the robot is operating in an environment where there are several small passages (gates), it is necessary that the generated reference points are as correct as possible.

- Speed and reliability
  As the algorithm should be implemented on an embedded processor, it is imperative that it is fast, and can run for extended periods of time without failing.

To accommodate for these demands a three module system is developed. The complete computer vision system is described by figure 12.2.

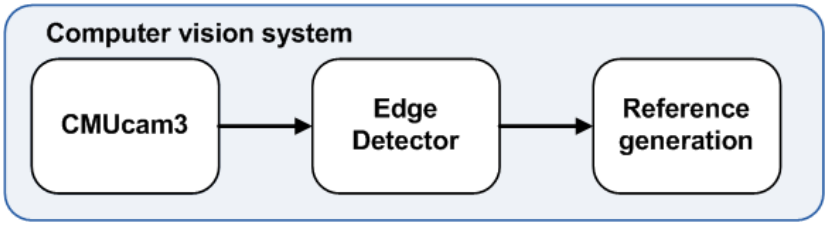

Figure 12.2: Shows the composition of the vision system of the MR

Each of these three modules of the vision system are shortly described below, and more thoroughly analyzed in the following sections. The lowest layer, the image acquisition block, is mainly composed of camera module CMUcam 3[20]. The CMUcam 3 is a CMOS camera sensor combined with a 60 MHz Arm 9 processor board. This entails that it is possible to conduct image processing directly on the camera board. The CMUcam3 includes an open

source development environment, with a lot of useful functions already implemented.
The next two levels, the edge detector and the reference generator are the two modules which should provide the MR with the features needed for the analysis of the image acquired by the CMUcam. This entails that they are implemented in software.
The first of these two blocks is the edge detector. This block provides a foundation on which more sophisticated algorithms can be run. It is a so called feature extractor. Feature extractors are algorithms that filter an acquired image for all unnecessary information. As the name suggests, an edge detector removes all information from the image but the location of edges. As several studies have shown that the human vision is composed of several types of edge detectors [32], it is proposed that the information derived from the edge detector, and the acquired image should be enough for both trajectory generation and other visual tasks, e.g. ball recognition.
The third and final layer is the reference generation. This layer uses information from the edge detector to robustly generate reference points of the line.
These three layers will in the following sections be described and analyzed.

12.1 CMUcam 3

This section will briefly describe the hardware composition of the CMUcam 3, what advantages the open source environment brings, and finally some details of the low level image acquisition.

12.1.1 Hardware and physical characteristics

The CMUcam 3 can be seen from figure 12.3, the physical dimensions are: 5.5X5.7 cm. From the Figure it can be seen that there are quite a few ports, and jumpers. For simplicity, one can describe the camera modules as composed from three vital components: a 60MHz microcontroller, a FIFO queue, and the CMOS camera sensor module. These are connected as seen from Figure 12.4. The microcontroller is used for image processing, and configuration of the camera. The FIFO is used for temporary storage, and finally the CMOS sensor provides the raw image. The CMOS sensor is able to provide images with a resolution of 352x288 pixels, but as described in 12.1.3 this is down sampled to provide fast image processing.
To interface the CMUcam, one needs to use serial communication, which is allowed with a speed of up to 115.200 kB/s.

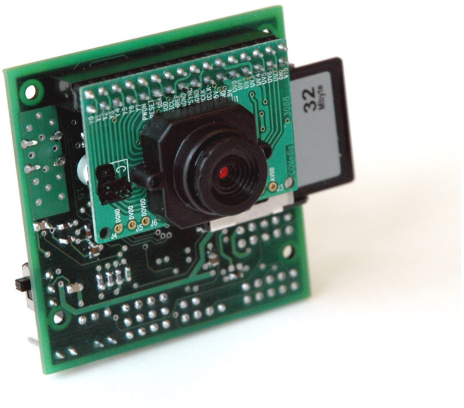

Figure 12.3: Shows a picture of the CMUcam 3, with mounted camera sensor

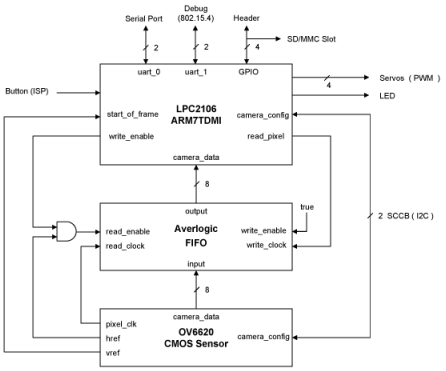

Figure 12.4: Shows the main hardware architecture of the CMUcam 3

12.1.2 Open source environment

The open source development environment provides a lot of useful functions and features for the developer, like the basic image manipulation library, which provides functions as:

- Arbitrary image clipping
- Image down sampling
- Threshold and convolution functions
- Software JPEG compression

One of the biggest advantages is that all drivers for the serial communication, the SD card, the LED's and CMOS sensor are already implemented, with clean and open interfaces. Another great advantage is the so called virtual cam environment. In this environment it is possible to develop code for the camera directly on a PC.

12.1.3 Image acquisition

The initial step of the image processing, is to acquire the image from the CMOS camera sensor. This is done by sending a signal from the microprocessor to the CMOS sensor that it is ready to process a new frame. When this is done, the sensor starts to write the image to the FIFO. It writes a matrix composed of 176X143 pixels each with three 8 bit values, describing the intensity of the red, green and blue elements of the image, to the FIFO. (The

camera is able to generate images of 352x286 pixels, but by using the lower resolution the sensitivity of each pixel is increased, and faster image processing is obtained.)
This matrix is then read row by row from the FIFO, and processed. As the implementation of the Canny detector is quite cumbersome, computational wise, the image is further down sampled in software to a resolution of 88x72 pixels. This is done to speed up the image processing. One should also see this in a context of that, with this resolution, a total of 88x72x3=19008 pieces of intensity information is still maintained within the picture.

12.2 Edge detector

In the following section the edge detector is described. There exists many different edge detectors, some more sophisticated and hardware demanding than others. According to a comparison between several edge detectors, the following applies:

[33, If the images to be analyzed are all fairly similar in content, or if the application allows for tuning the parameters of the detector for each image, then the best choice is probably to use a well-tuned Canny detector.]

The Canny edge detector was proposed in [34] by John Canny in 1986, and was meant as a way of improving the already existing edge detectors. Even though the Canny edge detector is not the fastest edge detector, it is often used as a common reference [33] when designing new edge detectors, and is thus well proven to yield good results. This entails that in this thesis it is chosen to base the edge detector on the Canny edge detector.

12.2.1 Algorithm outline

To be able to describe the Canny edge detector algorithm, one of the first things that must be analysed, is the concept of an edge. When working with edge detection, edges are defined as points in an image where the intensity changes greatly due to either change in colour, or due to object boundaries. When acquiring images from the CMOS sensor, the image will be affected by noise. This noise can create spurious edges that have no physical representation. This entails that it is necessary to find a very robust way of generating real edges and suppressing edges that are products of noise. This is what the Canny edge detector is all about. The algorithm is a step by step process consisting of 5 steps, which can be seen from Figure 12.5: When these five steps are applied to a raw image, only the image edges are left displayed in a boolean matrix. An example of an application of the Canny edge detection can be seen in [35] In this article the Canny edge detector is applied to images of a centrifugal

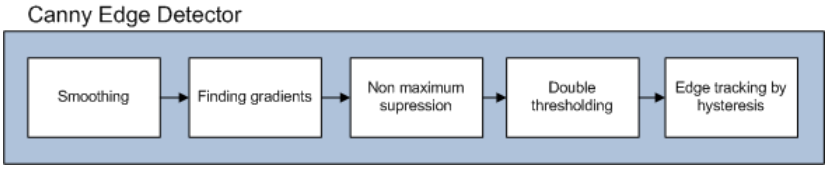

Figure 12.5: Shows the five steps of the Canny edge detector

pump (Seen on Figure 12.6) and the result can be seen from 12.7 From the picture, it is seen

Figure 12.6: Shows a picture of a centrifugal pump from Grundfos

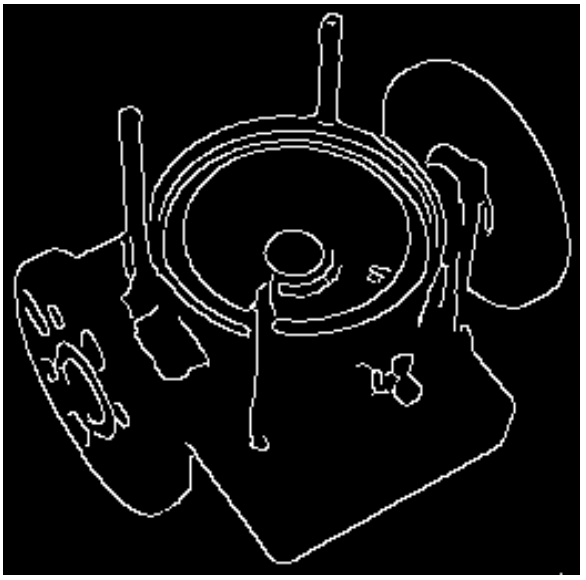

Figure 12.7: Shows the picture from 12.6 after the Canny edge detection algorithm has been applied

that only the edges are left of the picture, represented by white lines. In [35] this edge map is used for object recognition, which again demonstrates the applicability of the edge detectors. As described in section 12 the Canny edge detector should be implemented on the CMUcam. As the software development environment is based on c or c++, it is necessary to make an implementation of the edge detector in one of these languages. There exists free libraries for image processing, e.g. the open computer vision library [36], which is a c library with implemented Canny detection, but these are not developed for use on an embedded vision system as the CMUcam. This entails that an implementation is made, in c, which focuses on memory usage and speed. The five steps required for Canny detection are described below, with saved images after each step. These images are generated on the CMUcam and saved on the SD memory card, so they are actual images of what the camera sees. All the intensity information is based on intensity data from the red channel of the CMOS sensor. It has been chosen to utilize an image with a lot of insignificant objects on it, so that the robustness of the developed algorithm can be tested. The original image before the down sampling described in 12.1.3 can be seen from Figure 12.8 and after down sampling from Figure 12.9: From these figures it can be seen that the edges of the image have become somewhat less smooth,

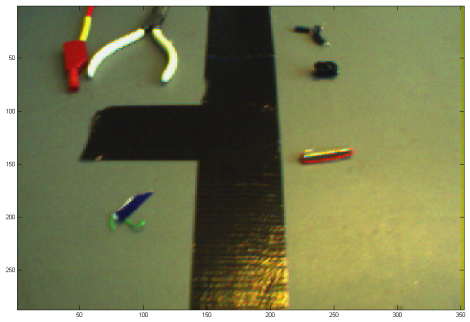

Figure 12.8: Shows a figure of a black line captured by the CMUcam before the down sampling is applied

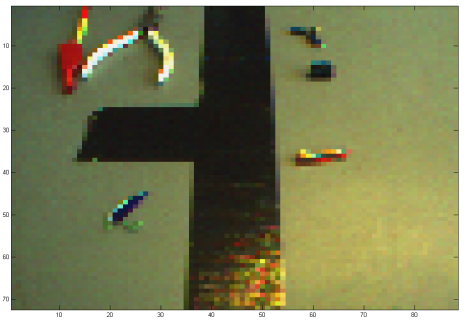

Figure 12.9: Shows a figure of a black line captured by the CMUcam after the down sampling is applied

but it is also seen that the general shape of the line is kept intact. Now that the image is down sampled, it is time to apply the steps of the Canny edge detector. An important observation from the images is the highlights on the black line. These are caused by reflections of the light in the laboratory which cause the black line to appear white.

12.2.2 Smoothing

The first step of the algorithm is to lowpass filter the raw image. When the raw image is obtained directly from the CMOS camera sensor, it is affected by noise, which can have a big impact on the further steps of the algorithm. To overcome this issue, Canny proposes to convolve the raw image with a 2D Gaussian filter. The 2D filter used in this implementation is the following, as it has proven sufficiently good:

$$G = \frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{12.1}$$

This is constructed from a Gaussian distribution with $\sigma = 1.4$, but this Gaussian filter could be extracted from any other Gaussian distribution. Notice however that the norming factor $\frac{1}{115}$ is omitted in the implementation, as divisions are computationally expensive. It is however significant that that one keeps in mind that there is a trade off between noise reduction and the ability to locate edge positions. If the red channel is extracted from 12.9, the characteristic found on Figure 12.10 is obtained. When the equation from 12.1 is used on Figure 12.10 the image from 12.11 is obtained.

From these images, several things are noticed. Firstly, as expected, a much more smooth looking image has been obtained; secondly it is noticed that due to the missing normalization,

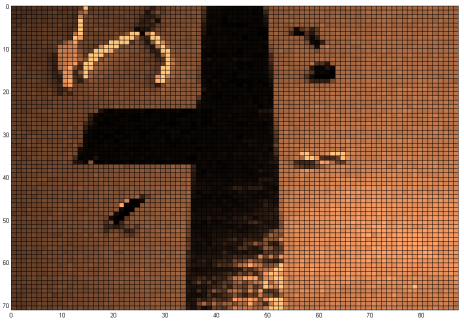

Figure 12.10: Shows a figure of a black line captured by the CMUcam when only the red channel of the image is used

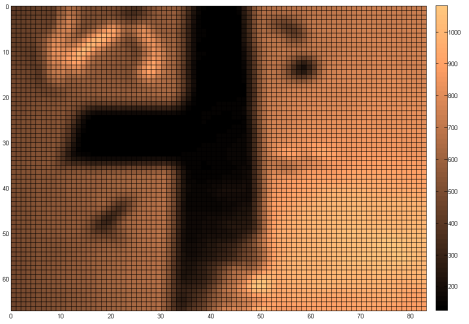

Figure 12.11: Shows the image of the black line from 12.10 when it is convolved with a Gaussian filter

the values of 12.11 are in a much higher range than in 12.10. This scale difference shouldn't affect the rest of the algorithm as long as the used thresholds are calibrated after these scaled values. It is also seen that even though the random objects placed next to the black line have been somewhat smoothed out, they are can still be seen on the final smoothed image. It is also seen that the reflections on the bottom of the black line are still present. Now it is time to take the next step, which is to derive the gradients of the image.

12.2.3 Finding gradients

In the second step of the Canny edge detector, areas with potential edges should be localized. As edges are defined as points where the intensity changes greatly, this entails that the intensity gradients of the image should be found. In the Canny edge detector, the gradient is approximated by means of the Sobel operators, which are two matrices that are convolved with the smoothed image. These operators can be seen from 12.2 and 12.3

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{12.2}$$

$$S_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{12.3}$$

The result of these two convolutions provides two pieces of information, the gradient intensity G, and the angle θ. If the resulting values of these convolutions are given by G_x and G_y, the intensity is derived by adding the absolute values of the convolution pixel by pixel, that is

$$|G| = |G_x| + |G_y| \tag{12.4}$$

As the S_1 operator can be interpreted as a differentiation along the x-axis and the S_2 along the y axis of the image, it becomes clear that the angle of the steepest gradient can be obtained by using the arcus tangens function.

$$\theta = atan2(G_y, G_x) \tag{12.5}$$

The angle of the gradient is used in the third step, the non maximum suppression, to point in the direction of increasing intensity. As there are only 8 neighbouring pixels, the returning angle of the arcus tangens function only needs to be able to return 8 different angles. 0,45,90,135,180... This entails that a lookup table is implemented, which returns the above 8 angles by analysing the ratio and sign of G_x and G_y. When these two operators are applied to the image of 12.11, the following two figures are obtained From these it is seen that all that

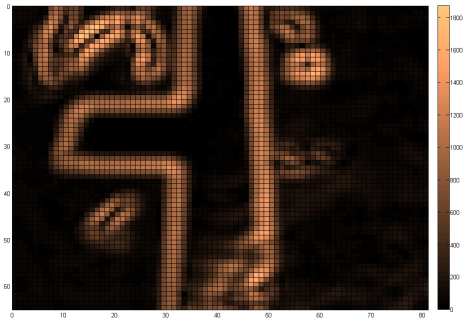

Figure 12.12: Shows the magnitude of the intensity gradient when the two Sobel operators from equation 12.2 and 12.3 is applied to Figure 12.11

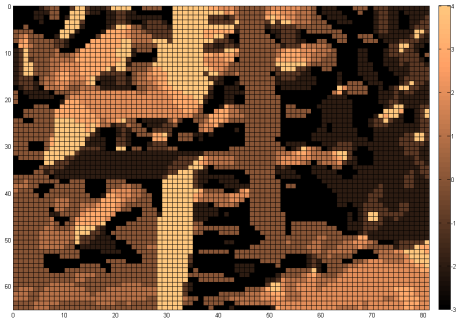

Figure 12.13: Shows the direction of the intensity gradients when two Sobel operators from equation 12.2 and 12.3 is applied to Figure 12.11

remains on Figure 12.12 is information regarding the edges of the picture. Note that not only the edges from the black line have been extracted, but also the edges of the random objects. In the bottom of the image it can be seen that the reflections have produced some irregular edges on the black line.

12.2.4 Non maximum suppression

The third step, called non maximum suppression is introduced to remove information which is irrelevant for edge detection. The derived gradient magnitude from Figure 12.12 gives information on the edges of the image, but these edges are very thick, and to determine exactly where the edge is located non maximum suppression is needed. In this step each point of the image is compared to the points in the direction of both the positive and negative gradient. If the current point is a local maximum no actions are taken. However if the point is not a local maximum, it is truncated to zero. This entails that all edges become only a pixel

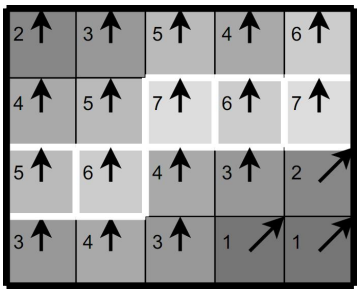

Figure 12.14: Shows a visualization of the third step of the Canny edge detector, the non maximum suppression

wide. The procedure is easily visualized by the example figure on 12.14, which is taken from [35] On this figure a section of an image is shown, with the intensity shown as a number in the upper left corner, and the gradient as an arrow. In the example, it is seen that almost all gradients point to the north, which entails that most of the pixels are compared to the pixels above and below. The pixels which turn out to be local maxima are marked by white edges. All other intensity values are truncated to zero. As this step in practice is conducted in the same sweep as the fourth step (Double thresholding) to increase speed, no images of this step can be seen.

12.2.5 Double thresholding

The fourth step is the double thresholding. After the step of non maximum suppression, each pixel is still labelled by their respective intensities. Many of those pixels that are not truncated to zero will be real edges. However some of these edges will be caused by the effect of noise or possibly rough surfaces. To remove these "false" edges, the Canny edge detector uses a double threshold. Two thresholds are chosen, one high, and one low. These can be relative to the maximum intensity of the image or determined beforehand. As the light sources on the obstacle course are always evenly distributed, fixed thresholds are chosen. If pixel intensity is larger than the high threshold it will be marked as a strong edge, if it is below the low threshold it will be truncated, and if it is in between the two thresholds it will be labelled a weak edge. When the double thresholding is applied to the sample image, the following is obtained.

On the figure several strong edges, marked by the value 1, and fewer weak ones, marked by the value 2, are present. It can be seen that many of the placed objects cannot be distinguished from the edges of the black line, simply because they are physical objects and therefore have a strong edge. It is however seen that the bolt on the right side of the black line only is marked by weak edges. It is also seen that the reflections have been marked as strong edges even though they are not physical objects. By studying 12.12 it can be seen that the gradient magnitudes of the reflections are just as large as the gradients of the edge of the black line.

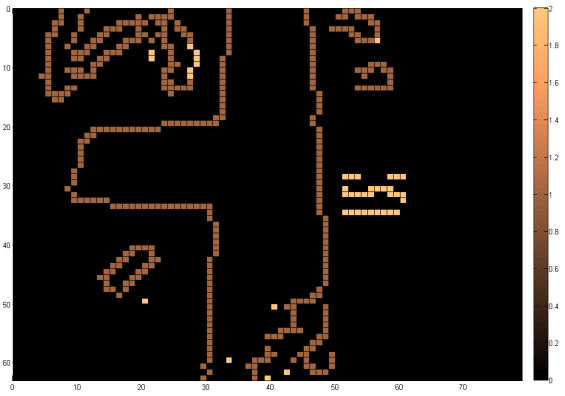

Figure 12.15: Shows the intensity gradient image of Figure 12.12 when non maximum suppression and double thresholding have been applied

This entails that this cannot be removed by simply changing the threshold.

12.2.6 Edge tracking by hysteresis

The final and fifth step of the Canny edge detector is the edge tracking by hysteresis. In this step, the weak and strong edges derived from the fourth step are used to determine the final edges of the image. It is assumed that all edges that are labelled strong edges are in fact edges of the image, and can immediately be included into the edge map. The next step is then to determine if the weak edges in fact are real edges or simply due to noise/colour variations. In the Canny edge detector, the weak edges are only included if they are connected to strong edges in the image. This entails that if noise should be present in the final edge map, it will have to either occur close to a strong edge, or be of intensity larger than the high threshold. When the fifth and final step of the Canny edge detection algorithm is applied to the sample image, the following is obtained: From this image it is seen that the weak edges of the bolt on the right side of the black line have completely vanished. It is however also seen that there remain several other strong edges. Most of these are from physical objects, but a small part of these are produced by reflections. This concludes the Canny edge detection. This edge map can now be used in combination to generate good references of where the line is located.

12.2.7 Summary of Canny edge detection

This section provides a quick summation of the steps involved in the Canny edge detector. In the previous sections five steps were conducted to transform an image taken from the

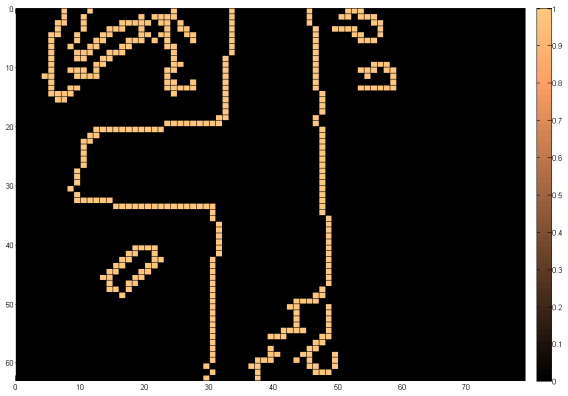

Figure 12.16: Shows the final edge map produced by the Canny edge detector. Notice that the weak edge in the upper right corner of Figure 12.15 has been removed as it is not connected to a strong edge

CMUcam into an edge map that can be used for the further navigation of the mobile robot. First the image (Figure 12.17) was filtered so that only the red channel was present (Figure 12.18), then it was smoothed by a Gaussian filter 12.19. Then the gradient magnitude (Figure 12.20) and direction (Figure 12.21) were found by means of the Sobel operators. Then the image was filtered by non maximum suppression and double thresholding in a single run(Figure 12.22), and finally the edge map was produced by edge tracking with hysteresis (Figure 12.23).

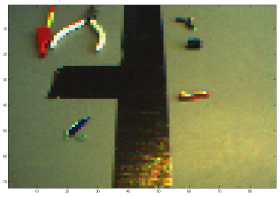

Figure 12.17: Original image

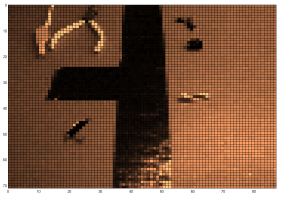

Figure 12.18: Red channel of original

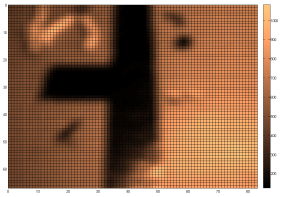

Figure 12.19: After Gaussian filtering

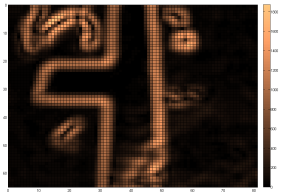

Figure 12.20: Gradient magnitude

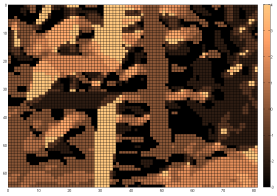

Figure 12.21: Gradient Direction

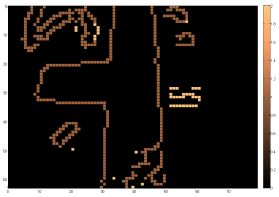

Figure 12.22: After double thresholding

With this part of the computer vision algorithm completed, a good foundation for trajectory planning and object recognition is established. The next section concerns the development

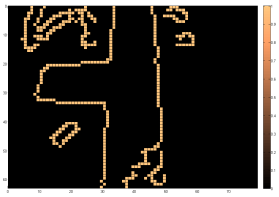

Figure 12.23: Final edge map

of a trajectory the MR should follow to stay on the line.

12.3 Reference generation using edge map

In the last section, an edge map of the CMUcam images has been produced. The objective of the following section is to use this foundation to be able to detect where the black line is, and to provide enough information to the on board computer, so that it can determine if the MR is at a crossroad.

12.3.1 Initial considerations

To be able to analyse and develop an algorithm for this, one needs to make an observation regarding the configuration of the observed lines. The black line from the obstacle course is long and never punctuated. This entails that there will always be a part of the line ending in the boundaries of the image matrix (If the line can be seen). With the possibility of dead ends, one part of the line can end in the middle of the image matrix. If the MR is tracking the line, the only places it could be wanting to go, is either to the endpoint of a dead end (a local minimum goal), or towards the edge of the image matrix (an image boundary goal).

- Image boundary goal
- Local minimum goal

It is however clear that the image boundary goals are the most common. This entails that in the further study of this thesis, only the image boundary goals will be considered.

Now that this initial delimitation has been made, it becomes clear that the reference points which the computer vision system should provide to the on board computer are the same as these image bounary goals. To find these image boundary goals, a divide and conquer approach, as the one used in the previous section is used. This entails that the algorithm is divided into five steps:

1. Clustering, and small cluster removal
2. Straight line connectors and remove false edges
3. Recreate junctions and thread construction
4. Small thread removal
5. Finding goal nodes

These steps will in the following be described further and applied to the sample image from figure 12.9.

12.3.2 Clustering and small cluster removal

The first of the five steps to produce the coordinates of the goal nodes is to sort and categorize the edge pixels found with the Canny edge detector. From the edge detector a binary map was extracted, where edge pixels were represented by the value 1 and non edge pixels by the value 0. To be able to determine which of these pixels are connected to which lines, it is necessary to utilize a clustering algorithm. According to[37, pp538] one can describe the clustering problem as one of finding natural groupings in a set of data. To be able to find natural groupings, one should first consider what a natural grouping is. That is, if a group of edge pixels should be defined as a natural group, what would be the common denominator. When observing the edge pixels from figure 12.16 it is seen that there are several natural groups. Some of these groups are part of the black line, and others are caused by reflections or various objects. The groups which describe parts of the black line can be characterised by two factors, they are directly connected in long patterns and they have a common direction. It is thus chosen that the following two similarity measures are used for clustering within a single group.

1. Distance between samples
2. Gradient direction of edge samples

To be able to conduct a clustering, the two similarity measures should be quantified into measureable parameters. By remembering that the Canny edge detector uses a double threshold to find edge pixels, and thus making it very robust against noise, and the fact that the edges generated from the black lines offer a very big contrast, it is chosen that the first similarity measure is quantified as: for a pixel to belong to a certain group of pixels, it needs to be 8-connected to at least one of the pixels in the group. A pixels p_1 is defined to be 8-connected to p_2 if it is located in one of p_1's 8 adjacent coordinates. The final similarity measure is the gradient of the edge samples. Remember that the gradient from the Canny edge detector is truncated to 8 values. It is thus chosen that a pixel needs to have the gradient point in a common direction to belong to a group. This also entails that if the line is curving, several clusters will be constructed on each edge. As the corners of objects will have a gradient different from the gradient of the objects edges, many small clusters containing only 1 pixel will be produced. To reduce the amount of spurious clusters, an extra filtration have been implemented to remove all clusters of less than 5 pixels. This should not affect the black line, as it is long and connected, but it should also remove parts of irregular shaped objects, such as reflections or random items lying close to the black line. When these two rules for clustering are applied to the edge map produced by the Canny edge detector, the figure from 12.24 is obtained.

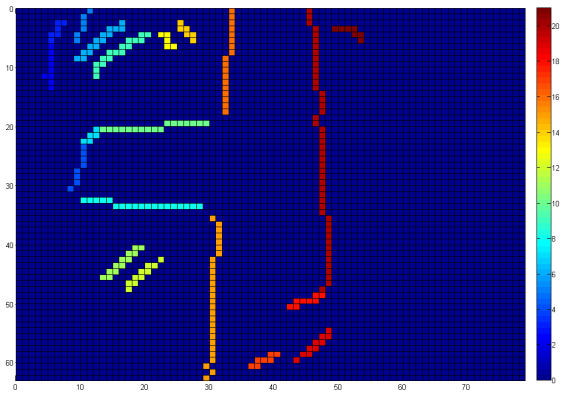

Figure 12.24: Shows the final edge map produced by the Canny edge detector when a clustering algorithm is applied to it.

From this it can be seen that 15 (2-16) clusters are obtained. One should note that the clusters start with the index 2. This is because the same matrix, used in the final steps of the Canny edge detector (which marked strong edges with the number 1), is reused due to memory constraints. It can be seen that no clusters with less than 5 members are present, which is as expected, and has two effects. The first being, that a lot of the edges, of the placed random objects, have been removed. The second effect is that in the corners of the black line there are suddenly gaps. The first effect is a good thing, as it reduces unnecessary information. The second effect is unintended, and should be accommodated, which is done in 4 "Recreate junctions".

12.3.3 Straight line connectors and false edges removal

Now that a set of large regular shaped edges has been produced it is time to make a more mathematical representation of the line edges. This is done by means of making a straight line approximation of the points in the clusters. Two methods for this straight line approximation have been considered. The first one is to use least squares to produce the best fit for a linear regression which should yield a mathematical expression like $y = a \cdot x + b$. The second method is simply to connect the extreme points (the endpoints) of the computed clusters, and connect these with a straight line. The second method is very dependent on the position of the endpoints with respect to the intermediate points. If the endpoints or intermediate points are affected by noise, the straight line acquired by this method might yield an inaccurate fit. However if the points in the cluster can be well approximated by a straight line, and the endpoints are not very noisy, a good fit should be obtained. The first method is however much more robust, and the linear regression will in most circumstances give a very good fit. However if the output is in the form of $y = a \cdot x + b$, it is impossible to

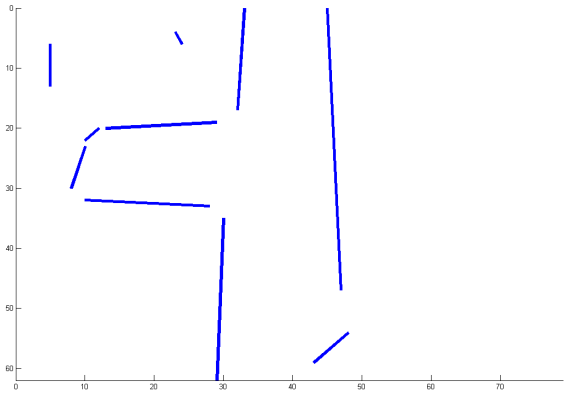

Figure 12.25: Shows the final edge map produced by the Canny edge detector when a clustering algorithm is applied to it.

represent vertical lines, and as the lines approach vertical, larger b and a values occur, which produces numerical uncertainties. As the lines detected are vertical when the robot is in the most optimal position, it is also in this position the first method would be the least accurate. Another element which could become critical is the time consumption. The linear regression is slower, and should use floating point operation, which is very slow on the microprocessor board. As the Canny edge detector should suppress most noise, and as the clustering ensures that all points of a cluster point in the same direction, and as the small cluster removal removes clusters that are either irregular or produced by noise, it is suspected that the second method should yield a good fit, while it does not suffer from the numerical uncertainties from the normal linear regression. This entails that the lines are approximated by connecting the extreme points of the clusters. As some of the clusters are made by reflections or the "wrong" physical objects, one more filtration step is applied. In this step, lines that are not black on one side and grey on the other will be suppressed. This results in, that lines resulting from e.g. shadows or discoloured are removed. In practice this is done by finding the middle point of the line connecting the extreme points of the cluster, and following a line perpendicular to the slope of the "cluster connector line", 5 pixels in each direction, and checking if a big enough change in intensity has occurred. By doing this to the image from 12.24 the image from Figure 12.25 has been found. From the image it is seen that many of the false lines have been removed, only one line from the reflections in the bottom of the black line, and two from the small pliers in the upper left corner.

12.3.4 Recreate junctions

Now that endpoints for all lines have been established, and all non black lines have been removed, it is time to recreate some of the information that was lost in the clustering process.

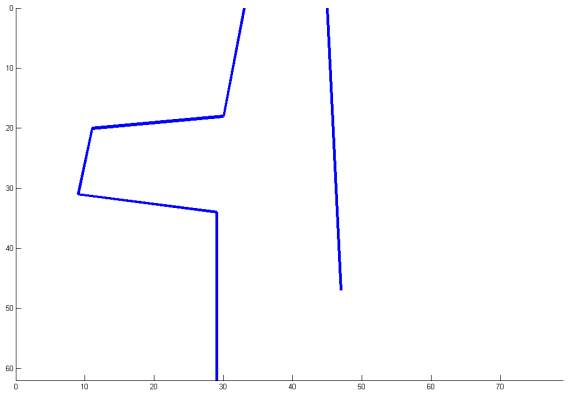

Figure 12.26: Shows the figure from 12.25 when the corners lost in the clustering are recreated, and with unconnected lines removed.

When using the small cluster removal in the section regarding clustering, there will be resulting gaps around corners of the black line. These originate from the fact that in these points the direction of the gradient will change and thus leave a couple of pixels forming a small cluster. To remove these gaps, it is checked if the endpoints of the computed straight line connectors are close to each other. If this is the case these endpoints are labelled as junctions, and the junction coordinate is found as the mean of the two endpoints. These junctions connect two nodes. After all nodes have been checked for junctions, the individual lines from the last step of the algorithm have become linked chains of cluster endpoints. If these linked chains of cluster endpoints are sufficiently long (span more than 20 pixels) they are kept, if not, they are labelled as unconnected and discarded. When this step is applied to the image of 12.26 the following is found.

From the figure it is seen that the three remaining false lines are removed and the only thing left is the outline of the black line. This entails that from an edge map describing the edge pixels of an image via the values 0 or 1, the contour of the black line has now been found with all unnecessary information removed. All that remains are connected points in space. The remaining problem has now been reduced to finding the possible destinations and make a trajectory to these.

12.3.5 Finding destination nodes

The final step is to determine the destination nodes from the set of endpoints and junctions. As described in section 12.3.1 two types of goals exists. But as the local minimum goal is not very uncommon it was chosen not to detect these. When this "chain" representation of the image has been obtained, finding the goal nodes can be seen to be quite easy. It is known

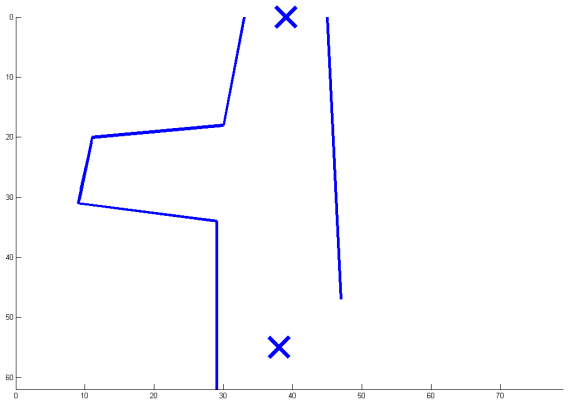

Figure 12.27: Shows the picture seen on 12.26 when the found destination nodes are plotted upon it

that all goals exists. Between two chain endpoints. That is, if there are three cluster chains, there will be three image boundary goal nodes. This entails that the chain endpoints with minimum distance are paired, and the mean value of these two endpoints are defined as the goal node.
When this last step is applied to Figure 12.26 two nodes are derived. When these are plotted it (Figure 12.27) is seen that they correspond nicely to the real endpoints of the black line This entails that a set of points, describing the endpoints of the line have been derived. Which can be used as a reference for linetracking.

12.4 Time consumption of the proposed algorithm

A small test of the time consumption of the proposed vision algorithm is made. This is done to verify that the data is produced at a reasonable fast interval. To test the time consumption of the proposed algorithm a small modification to the program is made so it reads out the consumed time on each of the steps of the algorithm. The obtained time in ms can be seen from table 12.1.

From this it is seen that the algorithm normally takes 79 ms to process an image, and derive the goals. It can however also be seen that the computation time changes with the provided image. The highest observed computation time is 90 ms, which entails that the camera is able to provide new information with a frequency of at least

$$\frac{1}{0.09s} = 11.1Hz \tag{12.6}$$

Now that a computervision system has been developed and implemented to solve the problem of perception, a small description of the problem of localization is made.

Description	Step number	Normal time[ms]	Max time[ms]
Image acquisition	1	40	41
Convolution with gaussian filter	2	17	18
Convolution with Zobel filter	3	12	12
Non maximum supression	4	6	10
Edge tracking	5	1	2
Clustering Algorithm	6	1	2
Straight line connections	7	2	2
Making junctions	8	>1	1
Making threads	9	<1	1
Finding goals	10	<1	1
Total		79	90

Table 12.1: Shows the time consumption of the various parts of vision algorithm

Chapter 13

Localization

In the following chapter the concept of localization is described. The main problem of local-

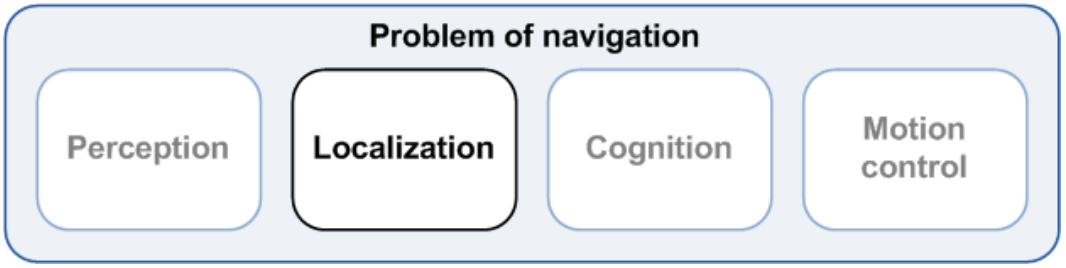

Figure 13.1: Shows the four sub problems of navigation. In this chapter the concept of localization is discussed

ization is to accurately derive where the robot is situated. This position can be given with a global frame of reference, such as the GPS system, or with a local frame of reference, such as a position given on a point in a map. As described in the section describing the delimitations of this thesis 3.2 the problem of localization will not be analysed in depth, but a small description of how a map is generated for the use of path planning is made.

The position should be derived in a certain context e.g. a map. To be able to do this, it is necessary to define how the representation of the surroundings should be conducted.

13.1 Representing surroundings

When representing the surroundings of the mobile robot, one can take two different approaches: to make a metric map, or to make a typological map. There are advantages and disadvantages with both approaches, some are listed below. And typically it is a trade off

between the degree of detail, and the degree of complexity. The main problem of typological maps is that normally a lot of information is lost in the transformation from a "real" representation to a typological one.

When regarding the obstacle course given from section 2 it is seen, that it is composed as a number of interesting points, where a task should be completed. These points are connected by a black line. Anything lying outside these black lines can be disregarded, as it provides unnecessary information. This entails that a good representation of the obstacle course can be derived from a typological map. And as path finding and other algorithms are easily implemented on typological maps, such a solution is chosen. Furtherer it entails that the map shown on figure 13.2 should be transformed to a typological representation. This is shown on figure 13.3. From figure 13.3 it is seen that a total of 17 nodes (0-16) are present with 24

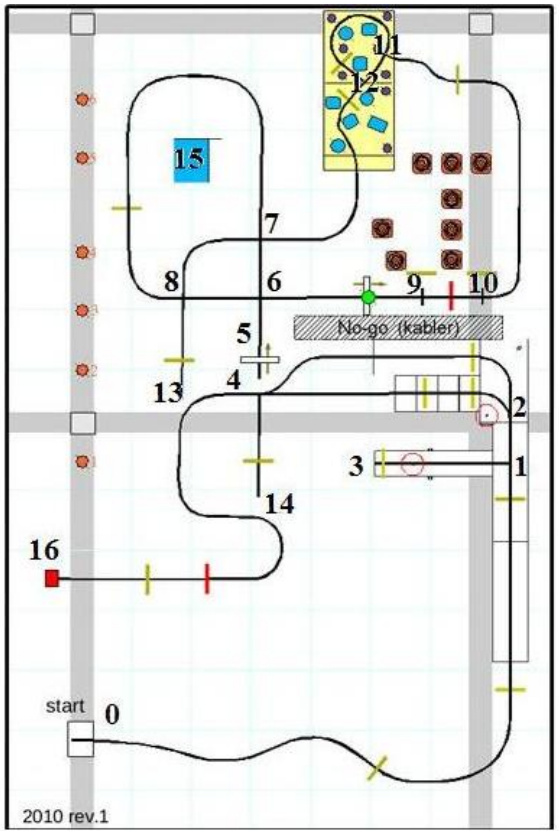

Figure 13.2: Shows the obstacle course on which the robot should operate as given from [12] but with added node numbers where crossroads or dead ends are present. Number 0 is the starting position, and 16 is the goal. The yellow bars represent point giving gates, the red bars represent penalty gates, and the two green ones represent respectively the start and the end of the racing track.

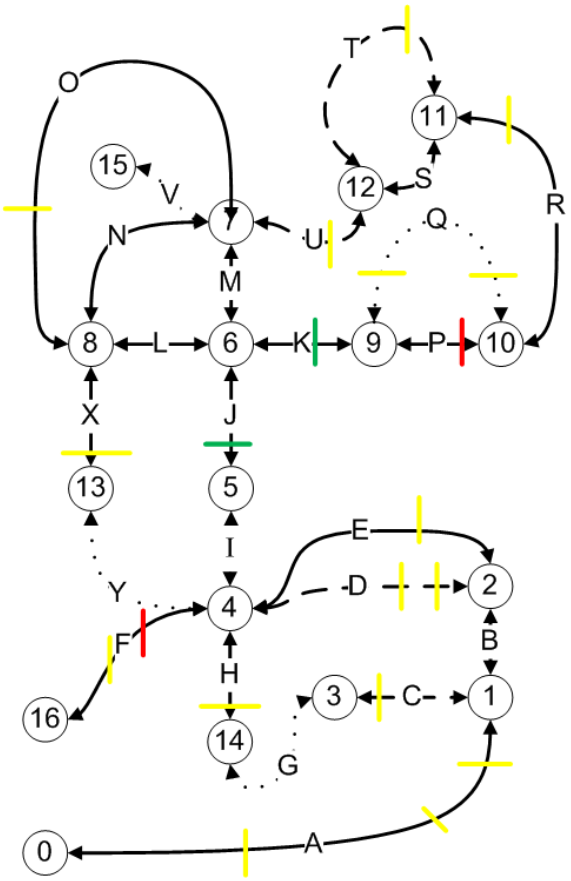

Figure 13.3: Shows the obstacle shown in figure 13.2, but in a completely typological manner. The numbers do, as before, represent the numeration of the nodes, and the letters represent the naming of the different paths. The lines with small punctuations show paths, which include moving without a line, and lines with big punctuations show paths, which require moving in difficult terrain (Slopes are not included)

paths connecting them in different manners (A-Y). These paths are divided into three subgroups: one that includes locomotion outside of the line (the lines denoted on Figure 13.3 with small punctuations); one that includes moving in difficult terrain (shown with big punctuations); and the normal paths (shown with unbroken lines). There are several parameters that can describe the transaction from one node to another by means of one of the paths A-Y. Such characteristics can be valuable when determining where the robot is located. These parameters could be the number of gates the robot has passed, the pathlength or any other feature of the path. These parameters will be discussed in section 14.4.1 where the problem of moving from node to node will be addressed.

13.2 Determining MR position

Now that a suitable environment, in which a position can be derived, has been defined, it is time to determine exactly how the MR should derive its position. As described in section 3.2 the problem of localization will not be treated in depth due to the timeframe of this project. To simplify concept as much as possible, the problem is reduced to simply updating the MR position, whenever the MR comes across a crossroad. That is, the MR moves from a point on the typological map to another point on the typological map. When a crossroad is reached, it is instantly assumed that the perceived crossroad is the correct node, and the MR position is made equal to the node position. This again entails that the MR position is limited to the set of node positions given by the typological map on figure 13.3.

Now that a small description of the concept of localization has been made, the focus is shifted towards MR cognition, that describes how the MR should act in a given situation to achieve its goals.

Chapter 14

Cognition

In the following chapter the concept of cognition is described. Cognition is the third segment of the problem of navigation (14.1).

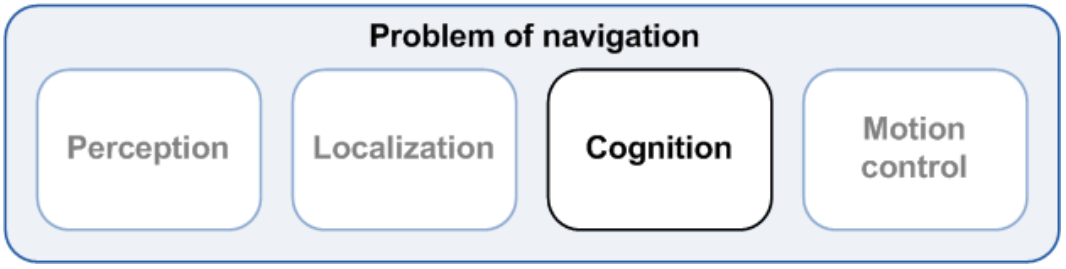

Figure 14.1: Shows the four sub problems of localization. The segment which is analysed in the following chapter is the concept of cognition

As the concept of cognition describes how a robot makes logical decisions about how it should achieve its goal. It is basically this block that should make the MR able to take decisions about what to do at what times. This entails that a system capable of using the obtained sensor data to navigate the MR around the obstacle course should be implemented.

14.1 Software system

It has been chosen to make a system composed of 5 blocks or layers. These 5 blocks can be seen from figure 14.2 From this figure it is seen that the top layer is the initialization layer. The next one is the goal management layer, followed by the path management layer, and the transfer management layer, and finally the motion management layer. These layers will be described in the following.

Figure 14.2: Shows the structure of the proposed system diagram. It can be seen that it is generally composed of 5 blocks, where each of them describes a new level of supervision.

14.2 Initialization

The first layer of the software system is the initialization. The structure of this system can be seen from figure 14.3 From this figure it is seen that the first things that happen are general

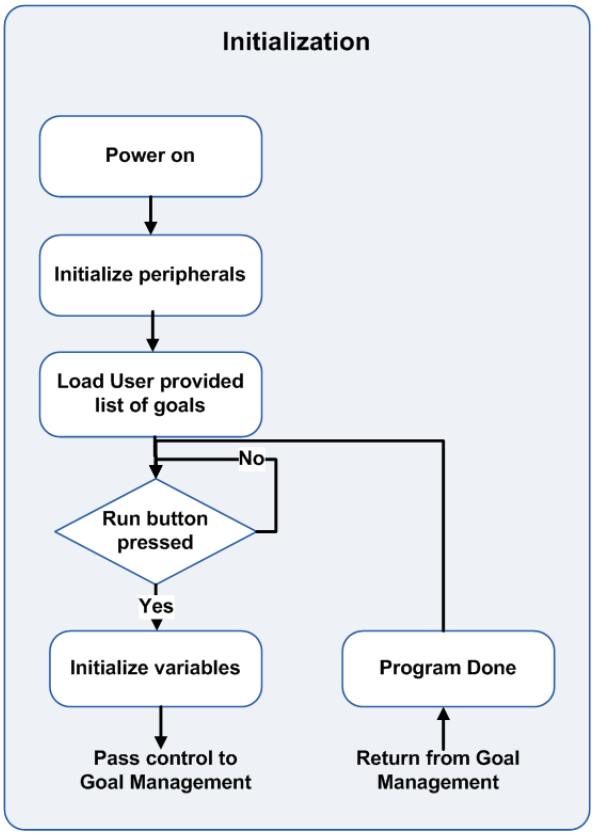

Figure 14.3: Shows the structure of the initialization procedure of the software system

start-up procedures. Afterwards a list of goals should be provided by the user of the system. Then the system waits for the user to press the run button, and then the initialization passes control to the goal management subsystem. When control returns, the MR should have completed all goals, and should have reached the final destination.
To ease the understanding of how these layers interact, a small example is used in the following sections. The example takes its beginning in the MR, standing at the starting point (node

0). Now the user commands the MR to move to node 8, 15, 11 and then the final position, node 16. This would entail that the goal list provided to the MR would be: [8 15 11 16].

14.3 Goal management

The next layer of the subsystem is the goal management layer. The structure of this layer can be seen from figure 14.4. This layer handles the accomplishment of the goals given in the

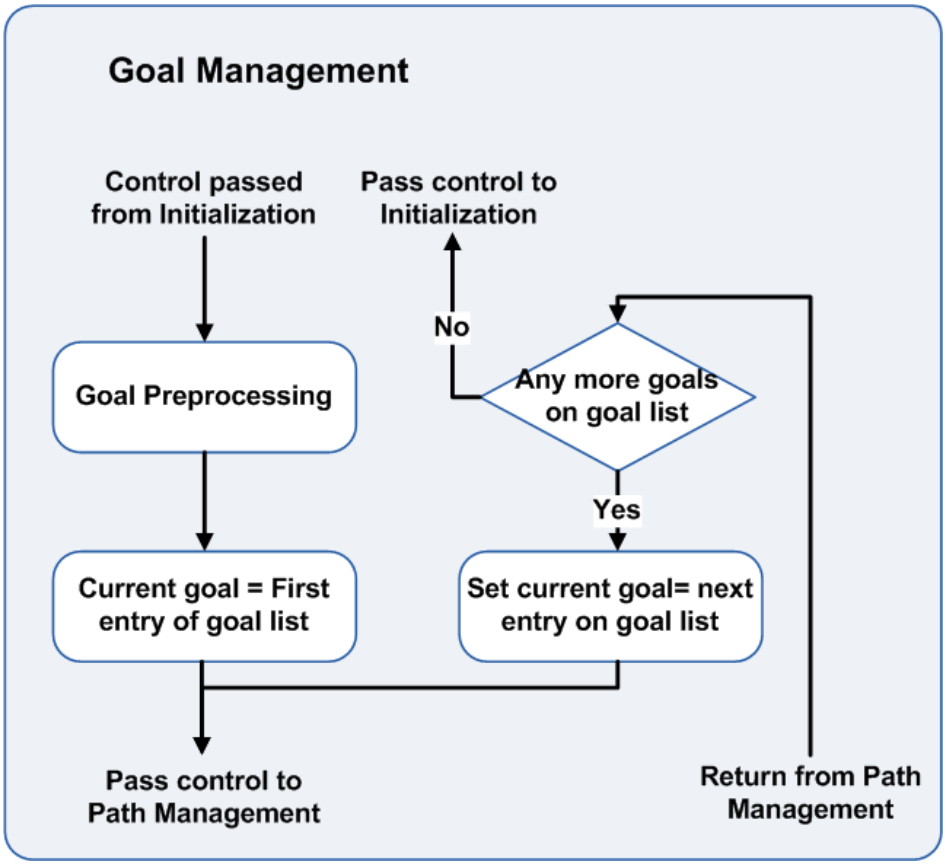

Figure 14.4: Shows the goal management layer of the proposed software system. This layer concerns the completion of the entire set of goals

initialization phase. When the layer is done, all goals have been accomplished. From this figure it is seen that control is passed from the initialization. Then some goal pre-processing will take place (described later), and then the first goal on the pre-processed list is chosen, and passed down to the path management layer. When control returns, the MR should have completed the goal, and if more goals are available, the next one is chosen. If none remains we have accomplished all goals, and the control is passed back to the initialization.

14.3.1 Goal pre-processing

As described, a block called goal pre-processing can be found on figure 14.4. In the following a small discussion regarding the contents of this block will be given. Generally the concept of goal pre-processing tries to address the following problem: given a map, and a set of goals, how should the MR move to successfully accomplish the given goals, while consuming a minimum amount of a certain resource e.g. energy or time?

Before a conclusion is reached, it is necessary to understand exactly what is meant by the concept of goals. A goal is a mission, as defined in section 2. There, it was defined, that the missions of the MR could be defined as two different types:

- Node missions
- Path missions

In the first type of missions the MR needs to get to a certain node, and do some sort of action e.g. moving into the tunnel. In the second type of missions, the path missions, the robot needs to travel along a given edge in the node map given in figure 13.3. However one might argue that one type of missions can be transformed to the other. As an example, a path mission dictating that path X should be traversed, could easily be transformed into two node missions, the first to node 13 and then to node 8 (or vice-versa). A node mission to node 13 could be transformed into a path mission dictating that edge X or Y should be traversed towards node 13. This entails that when the MR is given the goals, it should carry out, this list can easily be transformed into either a set of paths that should be traversed, or a set of nodes that should be visited. The way this interpretation is made, determines the type of pre-processing, and is thus important.

Goal pre-processing can be seen as the following. If a list of missions is given to the MR, should the MR take the goals and determine in which order these should be carried out, to reduce a cost variable (like time consumption, or power drain), or simply carry them out one after another? If this pre-processing should be carried out, the algorithm to do this would be dependent on what type of missions it receives, nodes or edges.
Generally one might state that, if the goals are represented by nodes, it can be defined in the following way: given a set of nodes in a graph, connect these nodes by the shortest possible route. One of the parameters that makes this problem hard to solve is the presence of directed paths in the graph (e.g. one cannot move up the tipping edge of the plateau). A solution to the problem could be to make a new graph of all the nodes that need to be visited, and interconnect all of these nodes by their shortest possible path in the original graph. If

this is done, the problem is reduced to a special type of "the travelling salesman problem" in an asymmetric map, since all the nodes should be visited in the shortest amount of time. However solving the travelling salesman problem is generally very slow process even though there exists heuristics for solving the problem in $O(n^2)$[38].
If the goals, on the other hand, are represented by edges, the problem can be described as: given a set of edges in a graph, traverse these edges via the shortest possible route. In the literature this problem is called "the rural postman problem" or RPP. Solving this is, as "the travelling salesman problem", a very slow process, but again heuristics for solving the problem exists [39]. This entails that both of these types of goals have a feasible approximately optimal solution.
On the other hand one must consider if such a pre-processing actually is beneficial for the mobile robot. It is clear that if an algorithm which sorted the misssion objectives were to be implemented, on one hand, it would provide the oppurtunity to save some time/resources, but on the other hand a degree of flexibility is lost. One needs to remember that the main task for the robot on the obstacle course is not to save time or resources (except at the race track), but to optimize the number of points it is able to collect. If an algorithm was to be implemented, there would be no control over the order in which the robot accomplishes the missions. There are situations where the time/resource optimization of the pre-processing algorithm could inhibit the point optimization, and this is problematic. For example when the robot reaches the goal node at node 16 it is forced to stop, that is, it is unable to earn more points. This final goal should always be accomplished as the last mission. Another example where, this time versus point optimization could cause a problem is, when there is a task that the robot often fails to accomplish successfully. It is often a good idea to save the tasks, in which the robot is error prone, to the end of the execution cycle. This is due to the fact that a failed mission accomplishment could mean that the robot gets lost or directly malfunctions.
This combined with the fact that there are no time constraints on the obstacle course, entails that no pre-processing of the provided goals is conducted. This again does not give an answer to what type of goal the MR should operate with internally.
However, as described in section 3, we need to implement an algorithm for finding the shortest path between two nodes in the graph if we get lost. As the same algorithm could be used for finding the shortest path between two goal nodes, the missions are presented as a list of nodes internally in the MR.

If we continue the example of section 14.2, where the MR was commanded to accomplish goals [8 15 11 16], the fact that no goal pre-processing is conducted, means that the goals will be visited in the exact order they appear on the list, that is from 0 to 8, from 8 to 15 and so on. The next layer is the path management layer.

14.4 Path management

The Path management layer is the third layer of the software system, and the structure of the path management can be seen from figure 14.5.

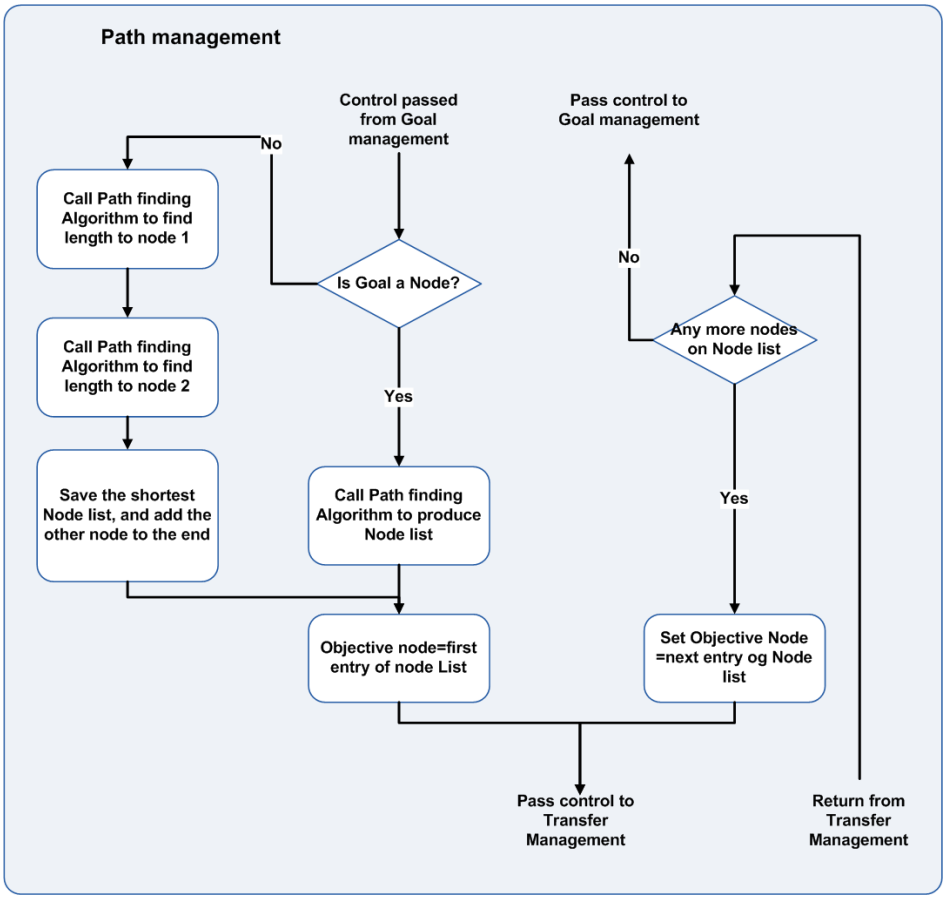

Figure 14.5: Shows the structure of the path management layer, which generally provides the MR with the ability to move from the current location to the desired location by the shortest path

The path management layer handles the transaction between goals. When this layer is done, the MR has accomplished the specified goal. First the control is passed from the goal management layer. As discussed in the last section, it can be called with two types of goals/missions, one concerning nodes, and the other concerning edges. On the left branch of the figure, when called from the goal management, an edge goal is divided into a node goal, and the shortest path is found, containing the specified edge. On the right branch, the shortest path to the goal node is found. Both of these branches call a path finding algorithm, which is described below. The path finding algorithm provides a set of nodes and transactions, which should be followed to move from the start node to the end node by the shortest path. When this list is found, the first node on this list is set to the current objective, and the control is passed to the transfer management layer. When the control returns from the underlying layer, the MR is at the new node. If there are more nodes on the node list, they are passed to the

transfer management layer, if not, the MR is at the specified goal, and control is returned to the goal management layer. In the following, the implemented path planning algorithm will be presented, and after this, the example, from the previous two sections will be continued.

14.4.1 Path planning

A path planning algorithm will in the following be an algorithm, which is able to tell the MR where to go to come from one node to another. The objective of the path planning algorithm is twofold. It provides the MR with the ability to get back to a node, if it has made a wrong turn, and makes the programming of the MR simpler, as only a set of goals, that needs to be accomplished, is needed to reprogram the MR. In section 3.2 it was chosen that no localization algorithm should be implemented, which again entails that the MR will have no means by which it can detect if it is lost or not. This means that, for this robotic system, only the second argument hold. However if it is accepted that the robot can become lost, one also needs to accept that, when the robot finally derives its true position again, it can be situated anywhere on the entire map. In the same way as the start position can be anywhere on the map, the goal position can be a node anywhere on the map. This entails that the set of start-endpoint positions contain:

$$\binom{17}{2} = \frac{17!}{2! \cdot (17-2)!} = 136 \tag{14.1}$$

pairs of positions. One could program a predefined path for each of these combinations, but it would be inefficient, inflexible and memory consuming. Instead a path planning algorithm is implemented.

There exist several algorithms for path planning, each with their strengths and weaknesses and areas of applicability. One can divide the approaches into two categories.
The first one is "local path planning", where the robot acts on behalf of sensor measurements, and has an incomplete understanding of the surroundings. Among these are some of the various "bug" algorithms, two of which are described in [1, p272-275] where only the general direction of the goal is known, and also the potential fields and dynamic window (see also [40]) approaches are presented. The local strategies do generally not require a lot of a priori information regarding the surroundings, and do not require heavy computation, which makes them fast and good for e.g. collision avoidance. As they only rely on a partial understanding of the map, they are often not able to produce optimal results, which leads to the second category.
In the second category, global path planning, the robot has an a-priori understanding of the

typology of the map, and is thus able to find an optimal route from one point to another. Among the last category of algorithms is the widely used A^* algorithm described in [2, p135]. As this algorithm can give an optimal solution (regarding path length) to the problem, and is easily implemented, when a typological map is known, it has been chosen to implement this algorithm.

It should however be noted that collision detection and avoidance is not incorporated in the A^* algorithm as a standard, and thus relies on a static map. As the obstacle course is static, it is concluded that it is unnecessary to integrate a collision avoidance scheme in the A^* Algorithm, even though several papers, that have been developed to the subject of integrating local path finding methods for collision avoidance with global methods, have been made. In [41] and [42] a unification of the local dynamic window approach and the global D^*(an algorithm much like A^*, but works better in a dynamic environment) algorithm are presented. In the following a description of the A* algorithm is presented:

14.4.2 A* Algorithm

In the following section, a short description of the A* (A star) algorithm is presented. For a more thorough explanation of the algorithm, good references are: [43]. The algorithm is a graph search algorithm, which is able to produce the shortest path between two nodes in a given graph. That is, if the MR is at node 1, and needs to get to node 6 in the graph from figure 13.3, the A* algorithm should be able to determine which nodes the robot should pass to get from node 1 to node 6 by consuming the least amount of resources. These "resources" are normally travelled distance, and thus the algorithm produces the shortest possible path.

Algorithm outline

In the following section, the basic outline of the algorithm will be presented, where after a small example will be shown to ease the understanding of the algorithm.
The first thing that should be introduced, when determining if a path is good, is to find a measure of the goodness of the path f. This would normally be the travelled distance.

$$f = g(n) \tag{14.2}$$

where:
f is the goodness of the path
g is the travelled distance
n is the nodes which has been passed

With this function it is possible to evaluate if the route was a good one or a bad one, but with this measure we need to try all possible paths to find out which one is the best one. This is due to the fact that there is no form of evaluating if you move in the wrong direction. You only know the distance from a start to goal basis. To improve this one needs to find another measure of goodness, one which can provide the distance information on a node to node basis. To do this a heuristic is presented. That is, the measure of goodness becomes the following:

$$f = g(n) + h(n) \tag{14.3}$$

Where:
h(n) is a heuristic of the remaining distance

With this new heuristic we get information about the travelled distance and the remaining distance for all nodes. Generally the A* algorithm can be described in pseudo code in the following way:

```
create a list, called the open list, initially containing only the starting node
create another list, called the closed list, which initially is initially empty
while (goal not reached) {
   consider the best node in the open list (the node with the lowest f value)
   if (this node is the destination) {
       we're finished
   } else {
      move the current node to the closed list and consider all of its neighbours
      for (all neighbouring nodes) {
      if (this neighbour is in the closed list and our current g value is lower) {
            update the neighbour with the new, lower, g value
            change the neighbour's parent to our current node
      }
      else if (this neighbour is in the open list and our current g value is lower) {
              update the neighbour with the new, lower, g value
              change the neighbour's parent to our current node
      }
      else this neighbour is not in either the open or closed list {
            add the neighbour to the open list and set its g value
      }
   }
  }
```

To ease the understanding of the above, the example used in the previous sections is continued in 14.4.2. To start the algorithm several parameters should be known. These are:

1. The start node

2. The end node
3. The resource consumed by travelling along a given edge (distance)
4. The coordinates of all nodes
5. A heuristic function for giving an approximation for the distance from any node, to the end node.

The two first required pieces of information, the start and end nodes are self explaining. The distance needed to travel along a given edge, and the coordinates of each node are both needed for calculating the goodness. The coordinates are used for the heuristic, and the travelling distance is used for the g(n) part. By analysing the map provided by the official RoboCup webpage [44] the following coordinates (Table 14.1) and distances (Table 14.2) are found:

The last of the needed five parameters is the heuristic. It is an approximation of the estimated movement cost to move from a given node on the graph to the final destination. The reason a heuristic is needed is, that the actual distance from the given node to the end node is unknown until a path have been found. For this heuristic, several types of metrics can be used, e.g. the Euclidian or Manhattan. These metrics affect the way the path finding algorithm is running.

- If the heuristic is 0, the measure of goodness is reduced to the travelled distance given in equation 14.2. The search algorithm opens nodes in all directions until the goal node is included in the search path. This entails that the algorithm is slow, but it is guaranteed to find the shortest possible path.

- If the heuristic is always smaller (or equal) to the actual distance, the A* algorithm is still guaranteed to find the shortest path, and with fewer opened nodes.

- If the heuristic is always completely equal to the remaining travelling distance, only the nodes on the optimal path will be opened. This is of course the optimal scenario, but as the exact distance from a given node to the goal is unknown, this is unfeasible.[1].

- Finally, if the heuristic is sometimes larger than the actual cost of moving from node n to the goal, the A* algorithm is not guaranteed to yield the best possible solution, but it can run faster.

[1] One could implement this by preprogramming all possible heuristics exact into the map.

Edge distances					
Edge	Distance [m]	Edge	Distance [m]	Edge	Distance [m]
A	12.1	I	0.5	Q	6.6
B	1.1	J	1.2	R	5.8
C	2.3	K	2.7	S	0.5
D	4.3	L	1.3	T	1.7
E	4.8	M	0.9	U	4.4
F	8.8	N	1.9	V	1.2
G	2.4	O	8.2	X	1.5
H	1.5	P	0.9	Y	1.4

Table 14.1: Shows the distance travelled when moving along a given edge.

Node coordinates			
Node	Coordinate (x,y)[m]	Node	Coordinate (x,y)[m]
0	(0,0)	9	(5.6 , 7.3)
1	(7.1 , 4.6)	10	(6.7 , 7.3)
2	(7.1 , 5.6)	11	(5 , 11.2)
3	(4.8 , 4.6)	12	(4.7 , 11)
4	(2.9 , 5.7)	13	(1.6 , 5.8)
5	(2.9 , 6.2)	14	(2.9 , 4.1)
6	(2.9 , 7.4)	15	(1.9 , 9.7)
7	(2.9 , 8.2)	16	(-0.5 , 2.8)
8	(1.7 , 7.3)	-	-

Table 14.2: Shows the coordinates of the nodes with node 0 as reference

As the optimal path is wanted, a heuristic which is always either smaller or exact the same as the remaining distance should be used. It has been chosen to use the Euclidian distance, as it represents the distance of a straight line from the current node to the end node, and thus is always smaller or equal to the actual travelled distance.

A* algorithm on obstacle course

Understanding the A* algorithm is much easier when applying it to an example. In the following, the A star algorithm is applied to the first goal of the example started in 14.2, where the MR is at start node 0, and needs to move to the nodes on the goal list [8 15 11 16]. In the goal management section, it was chosen that no pre-processing should take place, and thus, the first node to be visited is number 8. This entails, that the first time the path management layer is called, is to produce the shortest path between node 0 and node 8. The next time, it is to find the path from node 8 to 15, and so on. In the following, the steps involved in finding the shortest path between node 0 and 8 are presented

Initialization

In the initialization phase, the distances from node to node are applied to the search graph, and the start and goal nodes are identified. The start node is then added to the open list, and the f value is calculated

Open list				Closed list			
N.	g	h	f	N.	g	h	f
0(-)	0	7.5	7.5	-(-)	0	0	0
-(-)	0	0	0	-(-)	0	0	0
-(-)	0	0	0	-(-)	0	0	0
-(-)	0	0	0	-(-)	0	0	0
-(-)	0	0	0	-(-)	0	0	0

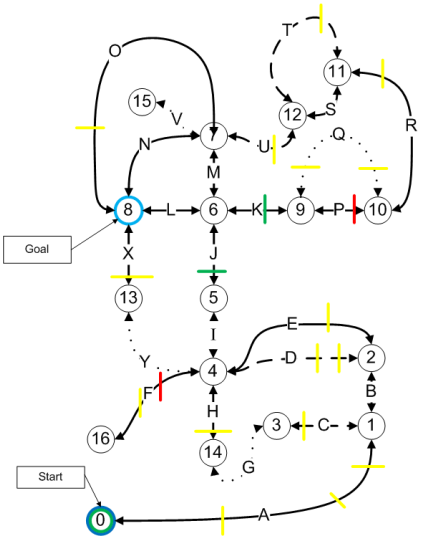

Figure 14.6: Shows the initial setup of the A star algorithm. with node 0 as start node, node 8 as end node (marked by blue), and node 0 marked as open (green)

Step 1

In the first step the node with the smallest f value is expanded so that all nodes, connected to the node (node 0), are put on the open list. As only the start node is opened, this is expanded, and thus node 1 is opened, the f value is calculated, and node 0 is appointed to be its parent node. Then node 0 is put on the closed list.

Open list				Closed list			
N.	g	h	f	N.	g	h	f
1(0)	12.1	6.0	18.1	0(-)	0	7.5	7.5
-(-)	0	0	0	-(-)	0	0	0
-(-)	0	0	0	-(-)	0	0	0
-(-)	0	0	0	-(-)	0	0	0
-(-)	0	0	0	-(-)	0	0	0

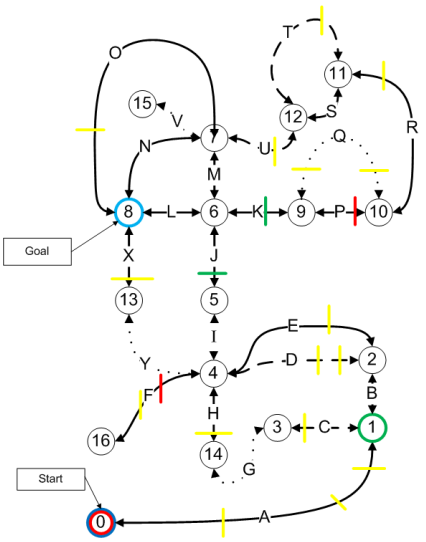

Figure 14.7: Shows step 1 of the A star algorithm, with node 0 as start node, node 8 as end node (marked by blue), node 0 marked as closed (red), and node 1 marked as open

Step 2

In the Second step again the node with the smallest f value is expanded so that all nodes connected to the node are put on the open list. As only node 1 is opened, this is expanded. There are three nodes connected to node 1, node 0,2 and 3. As node 0 is on the closed list, this is not opened, but 2 and 3 are. The f values are calculated, and node 1 is appointed to be the parent node. Then node 1 is put on the closed list.

Open list				Closed list			
N.	g	h	f	N.	g	h	f
3(1)	14.4	4.1	18.5	0(-)	0	7.5	7.5
2(1)	13.2	5.7	18.9	1(0)	12.1	6.0	18.1
-(-)	0	0	0	-(-)	0	0	0
-(-)	0	0	0	-(-)	0	0	0
-(-)	0	0	0	-(-)	0	0	0

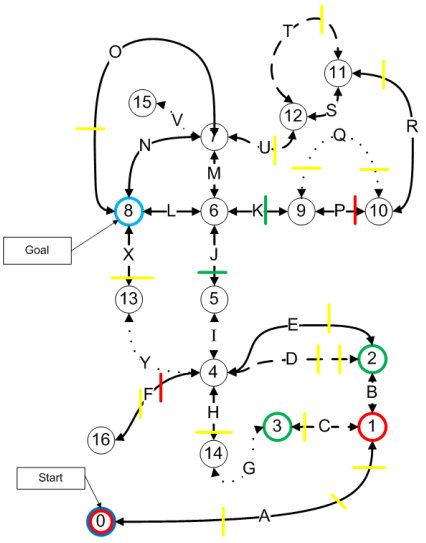

Figure 14.8: Shows step 1 of the A star algorithm, with node 0 as start node, node 8 as end node (marked by blue), node 0 marked as closed (red), and node 1 marked as open

Step 3

In the third step the same procedure as in step one and two is followed. Node 3 is expanded so that node 14 is put on the open list with node 3 as parent.

Open list				Closed list			
N.	g	h	f	N.	g	h	f
2(1)	13.2	5.7	18.9	0(-)	0	7.5	7.5
14(3)	16.8	3.4	20.2	1(0)	12.1	6.0	18.1
-(-)	0	0	0	3(1)	14.4	4.1	18.5
-(-)	0	0	0	-(-)	0	0	0
-(-)	0	0	0	-(-)	0	0	0

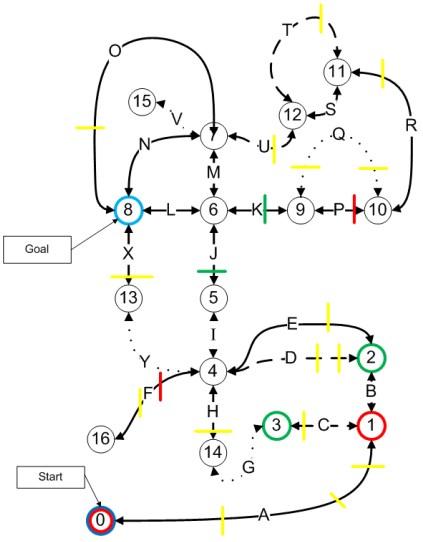

Figure 14.9: Shows step 1 of the A star algorithm, with node 0 as start node, node 8 as end node (marked by blue), node 0 marked as closed (red), and node 1 marked as open

Step 4

In the fourth step node 2 should be expanded, but the situation is somewhat different. It is seen that there are two paths connecting node 2 and 4, one shorter than the other. The first thing that happens is that node 4 is added to the open list, using one of the two connections, the g and h values are calculated, and the parent is identified (node 2). Then the node is added to the open list again via the other connection. As the node is already on the open list, a check is made to see if the current path to the node is shorter, than the path that added the node to the list (the size of the g value). Afterwards the g value and parent node are changed on the open list. If not, nothing is done. In this case path D is chosen.

Open list				Closed list			
N.	g	h	f	N.	g	h	f
4(2)	17.5	2	19.5	0(-)	0	7.5	7.5
14(3)	16.8	3.4	20.2	1(0)	12.1	6.0	18.1
-(-)	0	0	0	3(1)	14.4	4.1	18.5
-(-)	0	0	0	2(1)	13.2	5.7	18.9
-(-)	0	0	0	-(-)	0	0	0

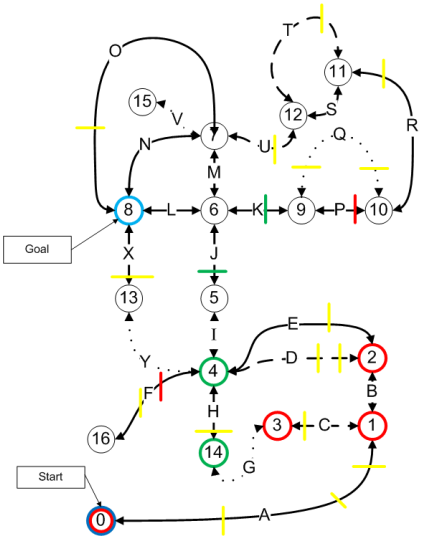

Figure 14.10: Shows step 1 of the A star algorithm, with node 0 as start node, node 8 as end node (marked by blue), node 0 marked as closed (red), and node 1 marked as open

Step 5

In the fifth step node four is expanded and 5,13 and 16 are added to the open list with node 4 as parent. Node 14 is already on the open list. This entails that it is checked if the path to node 14 is faster via node 4 than node 3. As this is not the case, nothing is changed in the open list entry of node 14.

Open list				Closed list			
N.	g	h	f	N.	g	h	f
14(3)	16.8	3.4	20.2	0(-)	0	7.5	7.5
13(4)	18.9	1.5	20.4	1(0)	12.1	6.0	18.1
5(4)	18.8	1.6	20.4	3(1)	14.4	4.1	18.5
16(4)	26.3	5	31.3	2(1)	13.2	5.7	18.9
-(-)	0	0	0	4(2)	17.5	2	19.5

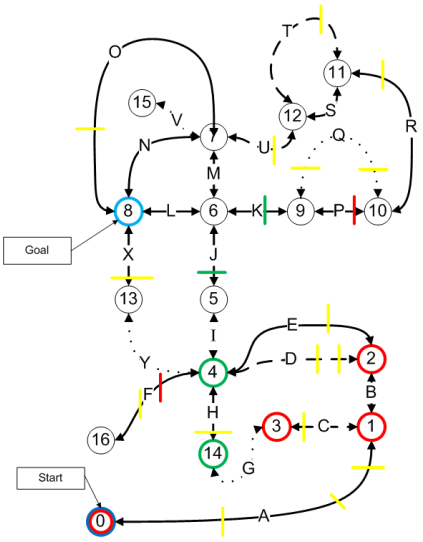

Figure 14.11: Shows step 5 of the A* algorithm

Further steps and backtracking

The procedure described in step 1 to 5 is continued until the node added to the closed list is the end node. When this happens the backtracking procedure is initialized. At this point, the algorithm has produced a closed list where all the nodes have a parent. The shortest path from the start node to each of these nodes, goes through these parent nodes. This entails that the shortest path from the start node to the goal node can be found by backtracking through the parent nodes, starting with the newly added goal parent. If the remaining steps of the algorithm are continued, the open and closed lists below are obtained. By backtracking from node 8 it is seen that we should go to node 13, and then 4 and so on. Finally this yields the shortest path to node 8, which is [0 1 2 4 13 8] with the length of 20.4 m

Open list				Closed list			
N.	g	h	f	N.	g	h	f
5(4)	18.8	1.6	20.4	0(-)	0	7.5	7.5
16(4)	26.3	5	31.3	1(0)	12.1	6.0	18.1
-(-)	0	0	0	3(1)	14.4	4.1	18.5
-(-)	0	0	0	2(1)	13.2	5.7	18.9
-(-)	0	0	0	4(2)	17.5	2	19.5
-(-)	0	0	0	14(3)	16.8	3.4	20.2
-(-)	0	0	0	13(4)	18.9	1.5	20.4
-(-)	0	0	0	8(13)	20.3	0	20.4

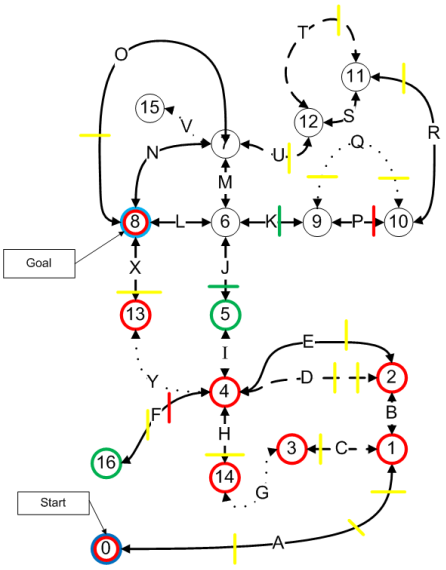

Figure 14.12: Shows a figure of the final situation when the A* algorith is applied.

14.5 Transfer management

The next layer presented, is the transfer management layer. This layer takes care of the movement from node to node, and the structure can be seen from figure 14.13.

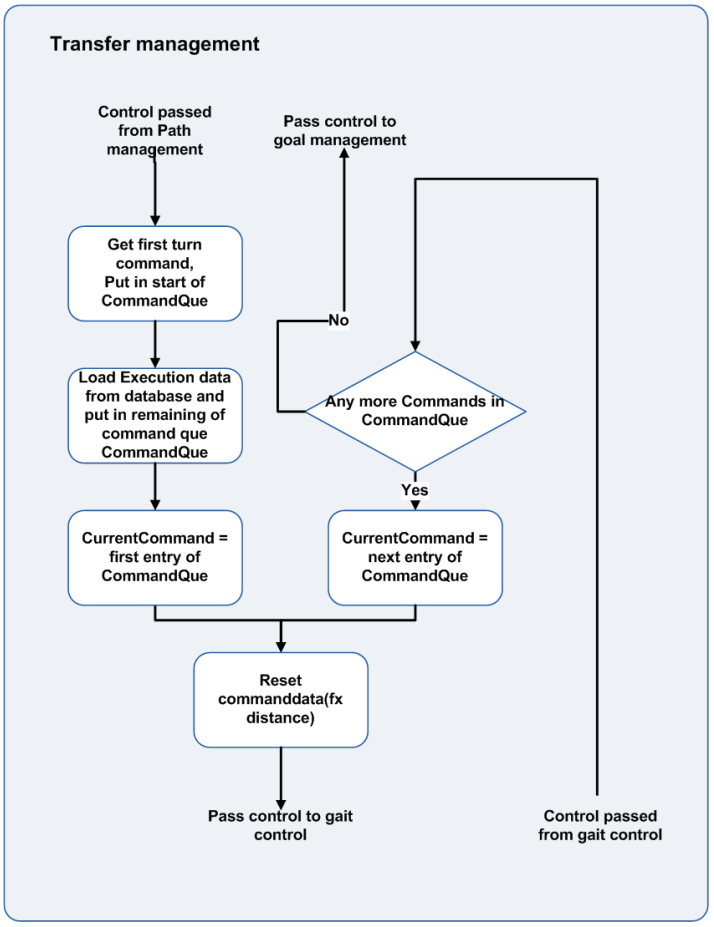

Figure 14.13: Shows the structure of the transfer management, which is the layer describing the transaction from one node to another

This layer is used to represent the various steps that need to be performed to come from one node to another. These descriptions of how the MR should move from one node to another should be preprogrammed, and should range from the very simple case like - move forward, until a crossroad is reached - to the more sophisticated, like if the MR should move from node 4 to 13. This procedure could involve several steps, like:

1. Turn to face the path towards node 16
2. Move 50 cm while following the line
3. Turn 70 degrees to the right
4. Move 50 forward without looking for the line

5. Move 50 cm forward, while looking for the line

6. Stop when on top of the line

This list of steps is called the command queue. A stop criterion is also passed on, which indicates when the next part of the command should be conducted. This stop criteria is called the command data, and can include travelled distance, lost line/found line, time, found crossroads.

14.6 Motion management layer

The final layer is the motion management layer. The motion management layer is the layer handling the movement through the passed transfers from the transfer management layer. It is this layer that tests if the MR has fulfilled the stop criterion provided by the command data. A simplified version of the diagram can be seen from figure 14.14. The real diagram can be found on the attached CD in the folder: SystemDiagram/SystemDiagram.pdf. Generally the

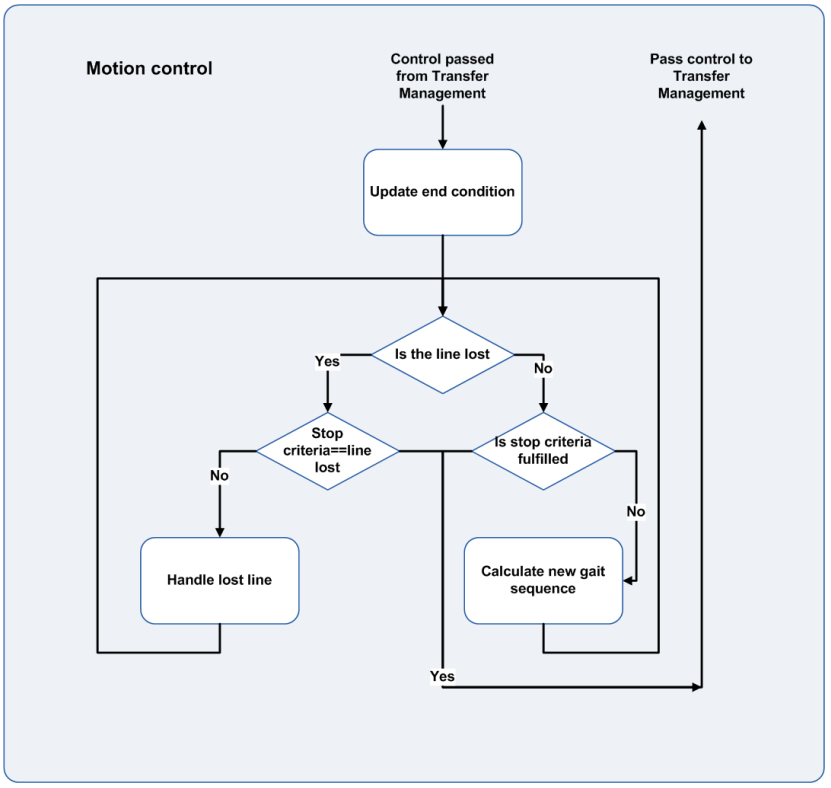

Figure 14.14: Shows the diagram for the motion management layer, which handles the transfers provided from the transfer management layer

main purpose of the layer is to constantly make sure the MR is on the right track. If the

line is lost, a series of steps is conducted to make the MR move in a way so that it can find the line again. If the stop criterion is fulfilled, it passes the control back to the transfer management, and if neither of these situations are true, it calculates a new gait on the basis of sensor measurements. It is in this level, that the actual control takes place, which will be described more thoroughly in part IV.

Part conclusion

In the previous sections, three main areas of the MR were analysed. The analysis were divided into four areas Perception, Localization, Cognition and Motion control. The last of which will be analysed in the following part of the report. A system capable of making the MR able to take descisions regarding how to find the fastest route between two points were derived. All drivers needed for the on board computer were written and verified, and large parts of the application were also implemented The code for these two applications can be found on the CD, under the folder /Software/CMUcam3/ and /Software/TS-7400Code.

Hardware issues

Unfortunately, the main microprocessor board, the TS-7400, stopped responding under the debugging phase of the implementation of the system. Despite countless attempts to make contact with it, no responces were obtained. This finally lead to the conclusion that a backup system should be implemented.
This backup were implemented in MATLAB, where a serial interface handled the communication with the CMUCAM3, and the SSC-32. The only implemented features of this system were the linetracking capabilities, which will be described in part IV of this thesis. And the computer vision system, described in section 12. As this section describes that crossroads should be handled on the on board computer, the computer vision system will however be limited to providing only two reference points.

Part IV

Control

The final part of this thesis, before the conclusion is the part concerning the control aspect of the mobile robot. If the terms of part III are to be used, one might say that the control problem can be said to concern the final level of the general problem of navigation, described in section 11.1. This problem is called Motion control see figure 14.15. In the following

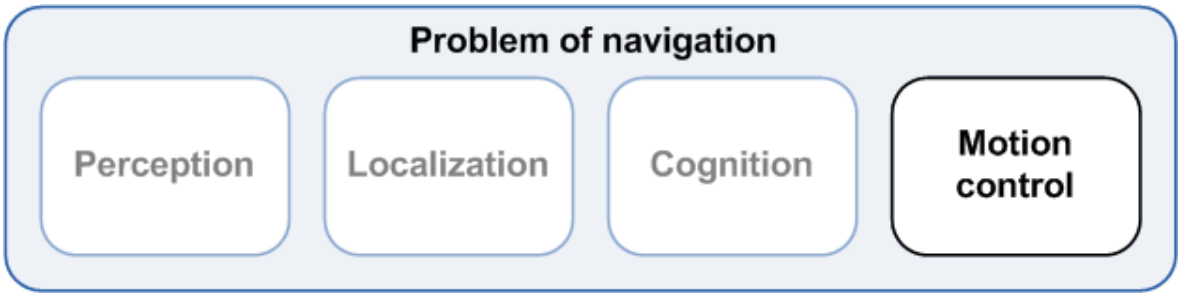

Figure 14.15: Shows the four sub problems of the general problem of navigation. In this chapter, the problem of motion control is adressed.

chapters, this problem will be analyzed. The derivation of how the robot should move, is decomposed into several sub problems. The first of these is to derive a real world reference from the points in the image, derived in section 12, concerning the computer vision system. When a useable reference for the control algorithm has been derived, a way of describing the gait of the MR, using three simple parameters is proposed. Then the constraints of the leg motion is considered and finally a control structure is proposed.

Chapter 15

Transformation from image to world coordinates

The first important aspect of the control strategy of the MR is to develop a control reference. From chapter 12 a series of points in image coordinates were given. As the MR is moving in real space, it is necessary to transform these image coordinates into something, which are physically meaningful. The following chapter will describe how the derived goal locations from the computer vision algorithm can be used as a reference in the real world. The goal location from section 12 is given as coordinates in the image. If this coordinate is named $\mathcal{P}_\rangle$, it can be expressed in image coordinates in the following way:

$$\mathcal{P}_\rangle = \begin{bmatrix} u \\ v \end{bmatrix} \tag{15.1}$$

It is clear that this point p is the image representation of another point $\mathcal{P}$ in the real world, that is, world coordinates. This point $\mathcal{P}$ can be written as:

$$\mathcal{P}_\rangle = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{15.2}$$

As the reference to the MR should be given in the real world, it is necessary to make a description of how the world representation of $\mathcal{P}_\rangle$ can be computed. This can however be quite problematic as a degree of freedom is lost when moving from the 3d world to the image plane. To be able to reconstruct this missing dimension, one needs use additional information regarding the position of points which are observed. There exists several ways of providing this extra information. One way to do this is by using two cameras(or more), that is using

a stereo vision system, this requires more hardware, and thus more power is consumed, it is also more expensive, but according to [2, 105] it is an attractive technology, as very few assumptions are required regarding the point positions. Several other approaches exists, like recovering depth from ego motion or from change of focus. In the first approach, several images are taken of an object from different locations, and by using knowledge about how far the MR has moved the missing dimension is derived. The second approach changes the focus, while taking several images, and from the blur circles, the dimension is reconstructed. One can say that none of these approaches utilise any assumptions regarding the position of the points, but are dependent on sufficiently good models of the motion of the MR(ego motion), or the focal length(depth recovery by focus). However good assumptions regarding the goal points of the MR can be made. As the goal nodes found from the computervision system can be said to be on the ground plane, in most circumstances, one can try to use this information to reconstruct the missing dimension. This method is referred to in [2] as "obtaining from ground plane assumption", and is based on a calibrated pinhole camera. In the pinhole mapping, it is assumed that all points in world space are mapped to the image plane through an infinately small opening. This pinhole provides a simple description of how an image is acquired, but does not take the effect of image distortion into account. According to [2, p86-88] this pinhole mapping can be described by means of a matrix $\tilde{\mathbf{P}}$, called the projection matrix.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \tilde{\mathbf{P}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{15.3}$$

The image coordinates u and v can be found as u=x1/x3, and v=x2/x3.
By insertions, and writing $\tilde{\mathbf{P}}$ as composed of three row vectors, the following is obtained:

$$\begin{bmatrix} u \cdot x_3 \\ v \cdot x_3 \\ x_3 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{P}}_1 \\ \tilde{\mathbf{P}}_2 \\ \tilde{\mathbf{P}}_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{15.4}$$

Thus the projection matrix describes the transformation from world coordinates to image coordinates. Finding this matrix is known as calibrating the camera. Generally this calibration is conducted by taking a picture of a number of points(at least six), with known real world coordinates, finding the image representation of these points, and then minimizing the mapping error in the least squares sense. This minimization have a solution, which gives the coefficients of the projection matrix In Appendix D this calibration is conducted and described in

greater detail. In this section $\tilde{\mathbf{P}}$ is found to be:

$$\tilde{\mathbf{P}} = \begin{bmatrix} 6.8336 & -17.7735 & 3.0099 & 138.6582 \\ -5.7147 & 1.2876 & 2.7192 & 395.7192 \\ 0.0347 & 0.0080 & 0.0226 & 1.0000 \end{bmatrix} \tag{15.5}$$

As described this matrix gives the image coordinates, when the world coordinates are known. The needed transformation, should however describe the inverse relation, the world coordinates from the image coordinates. If this transformation is assumed to be correct, then from equation 15.4, the following can be seen:

$$\begin{bmatrix} \tilde{\mathbf{P}}_1 \\ \tilde{\mathbf{P}}_2 \\ \tilde{\mathbf{P}}_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} - \begin{bmatrix} u \cdot x_3 \\ v \cdot x_3 \\ x_3 \end{bmatrix} = 0 \tag{15.6}$$

As x_3 can be described as:

$$x_3 = \tilde{\mathbf{P}}_3 \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{15.7}$$

One can rewrite equation 15.6 in the following way:

$$\begin{bmatrix} \tilde{\mathbf{P}}_1 - u \cdot \tilde{\mathbf{P}}_3 \\ \tilde{\mathbf{P}}_2 - v \cdot \tilde{\mathbf{P}}_3 \\ \tilde{\mathbf{P}}_3 - \tilde{\mathbf{P}}_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0 \tag{15.8}$$

Where row 3 provides no useful information, which again complies with the fact that more information is needed to compute the real world coordinates. As described, this information is provided by using the assumption that all points lie on a ground plane. This information is implemented by using the equation of the plane the points are situated upon. if A=[a_1 a_2 a_3 a_4] The equation of a plane can be described as:

$$[A] \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0 \tag{15.9}$$

If three points are known. p1=[x1,y1,z1],p2=[x2,y2,z2] and ,p3=[x3,y3,z3] these can specify the planar equation by means of the following determinants:

$$a_1 = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \quad a_2 = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix} \tag{15.10}$$

$$a_3 = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \quad a_4 = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix} \tag{15.11}$$

By using the first three points from which the camera were calibrated of appendix D, the following parameters are found:

$$a_1 = 0 \;\; a_2 = 0 \;\; a_3 = -10.82 \;\; a_4 = -162.3000 \tag{15.12}$$

If this plane equation is used in combination with 15.8 the following is obtained

$$\begin{bmatrix} \tilde{\mathbf{P}}_1 - u \cdot \tilde{\mathbf{P}}_3 \\ \tilde{\mathbf{P}}_2 - v \cdot \tilde{\mathbf{P}}_3 \\ A \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{B} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0 \tag{15.13}$$

If one partitions **B** into four column vectors,

$$\begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 & \mathbf{B}_3 & \mathbf{B}_4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0 \tag{15.14}$$

$$\begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 & \mathbf{B}_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = -\mathbf{B}_4 \tag{15.15}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = - \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 & \mathbf{B}_3 \end{bmatrix}^{-1} \mathbf{B}_4 \tag{15.16}$$

$$\tag{15.17}$$

Which entails that a description of the world coordinates have been derived, using only the calibration matrix, the image coordinates, and the assumptions that all points lie on the ground. To test if this mapping is correct, it is tested on the image coordinates from section D, to see if it yields the correct world coordinates. The results from this test can be seen from table 15.1

Point	Image coordinate[-]	Real world coordinate[cm]	Transformed coordinate[cm]	Difference[cm]	Distance derivation[%]
1	(179 102)	(31.3 0)	(31.0346 -0.3017)	(-0.2654 -0.3017)	-0.8433
2	(286 137)	(25.5 -8.4)	(25.0801 -8.6430)	(-0.4199 -0.2430)	-1.1931
3	(77 72)	(37.4 10.7)	(38.3527 10.9873)	(0.9527 0.2873)	2.5578
4	(114 102)	(31.3 5.8)	(31.3429 5.7875)	(0.0429 -0.0125)	0.1252
5	(273 67)	(37.4 -9.5)	(37.7493 -9.3715)	(0.3493 0.1285)	0.7968
6	(180 63)	(40.2 0)	(39.6237 -0.1325)	(-0.5763 -0.1325)	-1.4331
7	(181 35)	(47.8 0)	(47.8714 -0.0041)	(0.0714 -0.0041)	0.1494
8	(79 149)	(23.6 7)	(23.6029 7.4844)	(0.0029 0.4844)	0.5885
9	(270 197)	(18.6 -5.9)	(18.0399 -6.5703)	(-0.5601 -0.6703)	-1.6100
10	(70 52)	(44.7 13.4)	(44.1419 13.1720)	(-0.5581 -0.2280)	-1.2858
11	(0 189)	(18.8 12.6)	(18.4903 12.3704)	(-0.3097 -0.2296)	-1.7016
13	(175 159)	(22.2 0)	(22.2236 -0.2861)	(0.0236 -0.2861)	0.1145
14	(267 44)	(44.7 -10.4)	(43.7537 -9.5359)	(-0.9463 0.8641)	-2.4255

Table 15.1: Shows the difference between the real world coordinates, and the coordinates derived by means of a transformation between the image coordinates and the real world

From this table it is seen that good approximations have been achieved. It is seen that the biggest deviation is about .2.5 percent, with an absolute deviation of about 1 cm in both the x and y direction. From these measurements it is concluded that a transformation from image to real world coordinates have been conducted, and that via this transformation a position reference from the computervision system to the gait controller can be established. This leads to the next chapter, which concerns a way of describing the gait of the MR, with simple parameters.

Chapter 16

Legged locomotion

Before the derivation of how the MR should move is started, the problem of legged locomotion is presented. In the introduction of this thesis (chapter 1) a number of advantages of legged locomotion were introduced. Generally legged locomotion of robots tris to emulate the characteristics of legged animals. There are many ways in which an animal or robot can achieve locomotion, but normally locomotion is obtained by going through a series of steps, in a cyclic manner. In the sixlegged case, it is custom to discern between three gaits. The wave gait, the ripple gait and the tripod gait. In the wave gait, one leg is lifted from the ground at a time. This results in a slow movement of the body, but as five legs are always on the ground at a time. It provides the highest stability. The ripple gait can be said to be composed of two wave gaits, that is two legs are lifted at a time. The ripple gait provides higher speed, but is less stable. The final gait is the tripod gait. In this gait three legs are moved at a time. This gait provides the highest speed, but is also the less stable one.

It is however not an easy task to determine how the servo motors of the robot should turn in order to make the MR move by means of a tripod, or wave gait. In section 4, it was described that the MR consists of six legs, each controlled by three servo motors. This entails that the inputs to the system is composed of 18 angle references. Locomotion by means of any of the described gaits is only obtained if these 18 angle references are changed in a meaningful way. In this system, this is even more complicated, as these 18 servo references should make the MR able to track a reference provided by a black line on the floor. Instead of describing the gait as a big vector of servo angles, a representation using only simple parameters should be found. This problem is adressed in the following chapters

Chapter 17

Gait generator

In the this chapter, the process of generating a gait by means of three simple parameters will be described. As mentioned briefly in chapter 16 legged locomotion is achieved by changing the angles of the legs in a cyclic manner. The gait of the MR will make it move and turn, and it is thus adjustments of the gait which makes the MR track the reference provided by the transformation of the image coordinates derived in chapter 15.

Before the real derivation of the gait is conducted it is important to introduce some terms which will be utilised in the remaining of this part. After these terms have been introduced the formal derivation of the gait will be conducted.

17.1 Definitions of terms

Generally, a gait is simply a description of how the MR should move it's legs, which entails that several terms regarding the different ways a legged robot can move should be used. In [45] a terminology regarding the leg movements of a hexapod robot is used, which will be adopted in the following. These terms are:

1. Protraction
   The movement the leg makes when it moves towards the front of the body

2. Retraction
   The movement the leg makes when it moves towards the rear of the body

3. Power stroke
   The leg is on the ground where it supports the and propels the body. In forward walk-

ing, the leg retracts during this phase. Also called the stance phase or the support phase

4. Return stroke
   The leg lifts and swings to the starting position of the next power stroke. In forward walking, the leg protracts during this phase

5. Anterior extreme position (AEP)
   In forward walking this is the target position of the swing degree of freedom during the return stroke

6. Posterior extreme position(PEP)
   In forward walking this is the target position of the swing degree of freedom during the power stroke

To make a consistent description of the gait, this terminology is expanded with another term:

- Normap position(NP)
  The initial position of the legs of the MR, and the point from which new end point positions are derived.

17.2 Prerequisites for gait generation

Now that the general terms are defined, it is time to look closer at the task at hand. The main virtue of the gait is, as already mentioned, to provide a way of locomotion to the MR, and thus enabling it to track the black line of the obstacle course.

17.2.1 Choosing type of gait

The first thing that needs to be considered is what kind of gait should be generated. In chapter 16 three types of gaits were described. As the gait type is a tradeoff between stability and movement speed one needs to consider what is most relevant on the obstacle course. As there are only a few places on the obstacle course where there is uneven terrain, it is chosen to concentrate on a tripod gait as it is the fastest.
When using the terminology from section 17.1, one can describe the tripod gait as the following sequence of events:

1. Initial phase
   In the initial phase, the MR is standing with all legs touching the ground in NP.

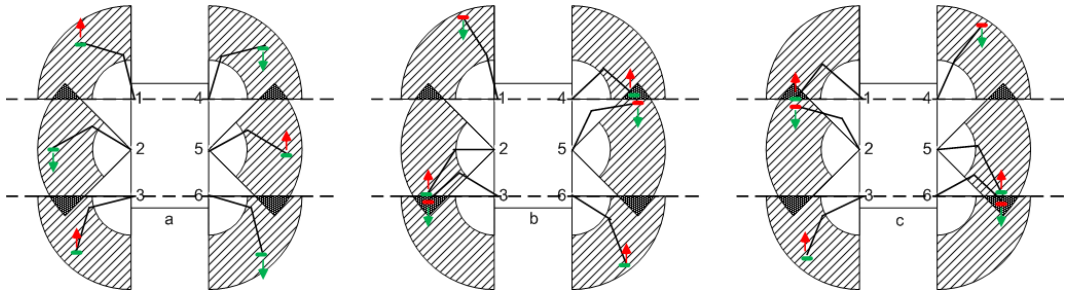

Figure 17.1: Shows a visualization of the tripod gait of the hexapod robot.

2. Liftoff
   Three legs are moved from the ground(the front and hind leg on one side, and the middle leg on the other)

3. Return and power stroke
   In this phase, two things happens at once, the three legs, which are not touching the ground are moved to their AEP(Return stroke), and the legs which are touching the ground are moved to their PEP (Power stroke).

4. Switch
   The supporting legs are switched so that the legs which were free in the air are now supporting the weight of the MR.

5. Return and power stroke
   Again the legs, which touches the ground performs the power stroke, and the ones in the air the return stroke.

6. Further steps
   Now steps 4 and 5 can be repeated, and locomotion is achieved.

This sequence can be visualized by looking at figure 17.1 On this figure, the Robot starts out by being in the initial phase(17.1 a), with all leg end effectors touching the ground(Marked with green). Then leg 1,3 and 5 are moved from the ground. Then 1,3 and 5 are moved to AEP, and 2,4 and 6 are moved to PEP(Figure 17.1 b). Then a switch between the supporting and free legs is conducted, and 2,4 and 6 are moved to AEP, and 1,3 and 5 are moved to PEP(Figure 17.1 c). On the figure, the coloured arrows describe the switches. As the gait type have been specified to be a tripod gait, the robot should have three legs in the air at a time when moving. How these legs are moved is the real problem of gait generation. This problem is divided into two subproblems. The the first being the problem of finding the position of the AEP and PEP points, and the second being the problem of how the transition between these two positions should be made.

These two problems will be considered independently in the remaining of this, and the following two chapters. In this chapter the position of the AEP and PEP points are found. Then in the chapter after this, the constraints of the AEP and PEP positions are found, and finally the trajectory between these two points will be considered.

17.3 Finding the AEP and PEP points

The first part of the problem of gait generation, is to find the AEP and PEP points of the legs of the MR. To be able to find these points, the locomotion is broken down into two parts, one concerning moving linearly relatively to the robot, and one concerning rotation.

17.3.1 Linear movement

The linear motion of the robot is the first concept treated. This type of movement is considered the normal case and the rotation will be described as a modifications to this part of the gait. The most straight forward way of defining the AEP and PEP is in relation to the normal position of the legs, when the MR is not moving. This normal position should be in the centre of the area, in which the leg is able to move. If the reference position of a leg end effector i is defined to be:

$$\mathcal{P}_i = \begin{bmatrix} xn_i \\ yn_i \end{bmatrix} \tag{17.1}$$

Then one could specify the general direction of the gait by means of an angle ϕ, which is an angular displacement from the x axis, and a speed indicator v, which describes how many centimeters per gait cycle the MR should move. This would yield a vector of the form:

$$\Delta\mathcal{P}_i = \begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix} = \begin{bmatrix} v \cdot cos(\phi) \\ v \cdot sin(\phi) \end{bmatrix} \tag{17.2}$$

One could then define the linear AEP and PEP in the following manner:

$$\mathcal{P}_{AEPL_i} = \mathcal{P}_i + \Delta\mathcal{P}_i = \begin{bmatrix} xn_i + \frac{v \cdot cos(\phi)}{2} \\ yn_i + \frac{v \cdot sin(\phi)}{2} \end{bmatrix} \tag{17.3}$$

$$\mathcal{P}_{PEPL_i} = \mathcal{P}_i + \Delta\mathcal{P}_i = \begin{bmatrix} xn_i - \frac{v \cdot cos(\phi)}{2} \\ yn_i - \frac{v \cdot sin(\phi)}{2} \end{bmatrix} \tag{17.4}$$

This entails that when determining the new placement of the legs during the return stroke, these always needs to be placed in their AEP positions, and likewise when determining the end position during the powerstroke, they needs to be placed in their PEP positions.

17.3.2 Rotation

The next part of the locomotion of the MR that needs to be considered is the rotation of the MR body. As already discussed, the rotation should be made as a modification of the linear motion derived in the previous section. In contradiction to the forward and sideways movement one cannot simply move the legs of the right and left side an equal amount, as this would not result in a turn, but in a translation. Instead the rotation is defined by making a rotation of the position of the legs around the centre of the robot. If one imagines a circle defined with the centre in the centre of the MR, and with each of the three legs touching a circle arc, to turn would indicate that each leg traverses a given distance around the circle edge.

As described in [14, p372], a rotation around the z axis, can be described by means of the following matrix:

$$R = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{17.5}$$

To give a rotation in the counter clockwise direction, the AEP should be rotated in the clockwise direction, and the PEP in the counter clockwise rotation. This entails that two different rotation matrices should be defined:

$$R_{AEP_i} = \begin{bmatrix} cos(-\frac{\theta}{2}) & -sin(-\frac{\theta}{2}) & 0 \\ sin(-\frac{\theta}{2}) & cos(-\frac{\theta}{2}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_{PEP_i} = \begin{bmatrix} cos(\frac{\theta}{2}) & -sin(\frac{\theta}{2}) & 0 \\ sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{17.6}$$

By applying these rotations to 17.4 and 17.3 the following two equations are found:

$$\mathcal{P}_{AEP_i} = R_{AEP_i} \cdot \mathcal{P}_{AEPL_i} \tag{17.7}$$

$$= \begin{bmatrix} cos(\frac{\theta}{2}) & sin(\frac{\theta}{2}) & 0 \\ -sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} xn_i + \frac{v \cdot cos(\phi)}{2} \\ yn_i + \frac{v \cdot sin(\phi)}{2} \\ z \end{bmatrix} \tag{17.8}$$

$$= \begin{bmatrix} cos(\frac{\theta}{2})(xn_i + \frac{v \cdot cos(\phi)}{2}) + sin(\frac{\theta}{2})(yn_i + \frac{v \cdot sin(\phi)}{2}) \\ -sin(\frac{\theta}{2})(xn_i + \frac{v \cdot cos(\phi)}{2}) + cos(\frac{\theta}{2})(yn_i + \frac{v \cdot sin(\phi)}{2}) \\ z \end{bmatrix} \tag{17.9}$$

$$= \begin{bmatrix} cos(\frac{\theta}{2})xn_i + sin(\frac{\theta}{2})yn_i + \frac{v}{2}cos(\frac{\theta}{2}) \cdot cos(\phi) + \frac{v}{2}sin(\frac{\theta}{2}) \cdot sin(\phi) \\ -sin(\frac{\theta}{2})xn_i + cos(\frac{\theta}{2})yn_i - \frac{v}{2}sin(\frac{\theta}{2}) \cdot cos(\phi) + \frac{v}{2}cos(\frac{\theta}{2}) \cdot sin(\phi) \\ z \end{bmatrix} \tag{17.10}$$

$$= \begin{bmatrix} cos(\frac{\theta}{2})xn_i + sin(\frac{\theta}{2})yn_i + \frac{v}{2}cos(\frac{-\theta}{2} + \phi) \\ -sin(\frac{\theta}{2})xn_i + cos(\frac{\theta}{2})yn_i + \frac{v}{2}sin(\frac{-\theta}{2} + \phi) \\ z \end{bmatrix} \tag{17.11}$$

Where the last equation is derived by means of the trigonometric addition formulas. Similar equation can be derived for $\mathcal{P}_{PEP_i}$, which gives:

$$\mathcal{P}_{PEP_i} = \begin{bmatrix} cos(\frac{\theta}{2})xn_i - sin(\frac{\theta}{2})yn_i - \frac{v}{2}cos(\frac{\theta}{2} + \phi) \\ sin(\frac{\theta}{2})xn_i + cos(\frac{\theta}{2})yn_i - \frac{v}{2}sin(\frac{\theta}{2} + \phi) \\ z \end{bmatrix} \tag{17.12}$$

These equations describe the end effector positions, both in the PEP and AEP positions. By Using these equations, locomotion, both linear and rotational can be achieved.

This entails a way of describing the extreme positions of the end effector has been developed. If a trajectory between these points were knoen, it would entail that the tripod gait could be described by means of three parameters ϕ, θ and v, instead of a large vector of servo motor angles. It is however clear that since the legs of the MR are not indefinately long it is not possible to simply place these AEP and PEP points anywhere. This leads to the next chapter, describing the constraints of the position of the legs.

Chapter 18

Constraints of AEP and PEP

As described in the previous section, it is not possible to place the AEP and PEP points in any location. The position could be out of reach for the end effector, or another leg could occupy the space. To remove the chance of collisions and to solve the problem of requesting end effector positions that are out of reach, an analysis of the constraints of the system is made. Generally two kinds of constraints are considered. The constraints affecting the position of the AEP and PEP, and the ones affecting the transition between them. These constraints can be divided into three categories. The constraints which affects the position of the AEP and PEP points, which are.

- Configuration space constraints
- Leg collision constraints

The constraints affecting the transition between AEP and PEP

- Kinematic constraints

And the constraints affecting both the position and transition between AEP and PEP

- ZMP constraints

In this chapter only the constraints applying to the position of the AEP and PEP will be considered.

18.1 Configuration space constraints

The configuration space of the legs of the MR is defined to be the points which the MR is able to reach, given that the MR is a given height above ground z. The configuration space can be described as composed of three different constraints, a minimum reach, a maximum reach, and an angular constraint. These constraints narrows the amount of possible end effector positions to the points of a section of an annulus. A visualization of this constraint can be seen from Figure 18.1. From Appendix E it was discovered that the maximum reach was

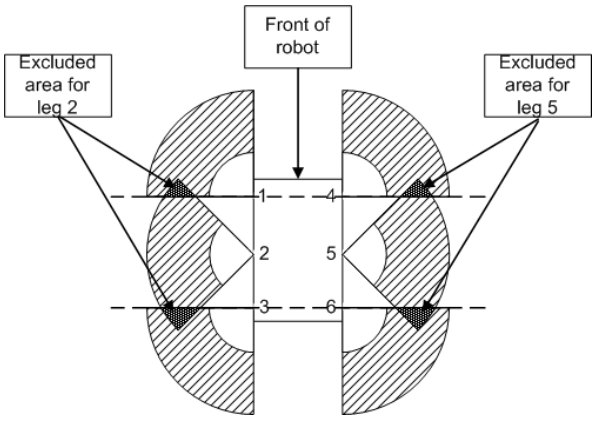

Figure 18.1: Shows the true constraints of the MR, and the imposed constraints to avoid leg collisions

found to be 17 cm, and the minimum 1 cm. The angular constraint is directly determined by the range of the servo motor, which is 180 degrees.

18.2 Leg collision constraints

The second constraint imposed on the MR is the Leg collision constraint. If the geometry of the placement of the legs on Figure 18.1 is considered, it is seen that reachable areas of the legs intersect. It can thus be concluded that the legs might collide if their complete configuration space were used. To avoid this a constrained region should be defined. This constraint is defined by dividing the area around the MR into 6 square regions one for each leg, where a leg is only allowed to move within its own square. This effectively removes the need for other types of leg collision avoidance, but removes a part of the configuration space, which is of course unintentional. But as the area lost to this configuration space division, is small compared to the total configuration space, and the fact that in a tripod gait, the legs would collide if the whole configuration space were used, only a small effect on the resulting gait is expected. The constrained space can be characterized by only two values, which gives the maximum and minimum x value of the middle legs. It has been chosen to allow the middle legs to attain a maximum x value of 9.5 cm and a minimum of-9.5 cm.

18.3 ZMP constraints

The final constraint which should be considered is the constraint conserning the stability of the system. This constraint affects both the position of the PEP and AEP, but also the trajectory between these. If the MR is to perform a dynamic stable walk, the ZMP should at all times be located inside the POS. This constraint is however treated in the next chapter concerning the trajectory between the AEP and PEP.

18.4 Mathematical formulation of constraints

As described in the previous section the endpoint positions of the MR legs are subject to two constraints. The first being the configuration space constraints of the legs, and the second being the confined areas described in section 18.2 for each leg. In the following these two constraints are treated analytically. The first constraint that is treated is the constraint made to avoid leg collisions.

18.4.1 Leg collision constraints

This constraints can be described as a constraints in the movement of the legs in the x direction, that is, the legs are subject to the constraint:

$$x_{i_{min}} < \mathcal{P}_{PEP_i}x < x_{i_{max}} \tag{18.1}$$

$$x_{i_{min}} < \mathcal{P}_{AEP_i}x < x_{i_{max}} \tag{18.2}$$

if the expressions for $\mathcal{P}_{PEP_i}x$ and $\mathcal{P}_{AEP_i}x$ from 17.11 and 17.12 are inserted into 18.1 and 18.2, the following is obtained:

$$x_{i_{min}} < cos(\frac{\theta}{2})xn_i + sin(\frac{\theta}{2})yn_i + \frac{v}{2}cos(\frac{-\theta}{2} + \phi)x < x_{i_{max}} \tag{18.3}$$

$$x_{i_{min}} < cos(\frac{\theta}{2})xn_i - sin(\frac{\theta}{2})yn_i - \frac{v}{2}cos(\frac{\theta}{2} + \phi) < x_{i_{max}} \tag{18.4}$$

This entails that a mathematical formulation of the constraint has been developed.

18.4.2 Configuration space constraints

The mathematical formulation of configuration space constraint can be found by a similar approach. If the position of the coxa joint for leg i is named.

$$\mathcal{P}_{cox_i} = \begin{bmatrix} x_{cox} \\ y_{cox} \end{bmatrix} \tag{18.5}$$

The constraint describing the configuration space can mathematically be described in the following way:

$$l_{i_{min}} < ||\mathcal{P}_{AEP_i} - \mathcal{P}_{cox_i}|| < l_{i_{max}} \tag{18.6}$$

$$l_{i_{min}} < ||\mathcal{P}_{PEP_i} - \mathcal{P}_{cox_i}|| < l_{i_{max}} \tag{18.7}$$

$$\theta_{min} < \theta/2 < \theta_{max} \tag{18.8}$$

$$\theta_{min} < -\theta/2 < \theta_{max} \tag{18.9}$$

$$\tag{18.10}$$

Where the first two equations describe the constraints posed by the length of the leg, and the two last ones describing the constraints posed on the angle the leg is able to reach. These first two constraints can be rewritten by using the equations describing the PEP and AEP positions obtained in chapter 17. If these calculations are conducted, the following equations are found.

$$l_{i_{min}} < \sqrt{\left(cos(\frac{\theta}{2})xn_i + sin(\frac{\theta}{2})yn_i + \frac{v}{2}cos(\frac{-\theta}{2} + \phi) - x_{cox}\right)^2 + \left(-sin(\frac{\theta}{2})xn_i + cos(\frac{\theta}{2})yn_i + \frac{v}{2}sin(\frac{-\theta}{2} + \phi) - y_{cox}\right)^2} < l_{i_{max}} \tag{18.11}$$

$$l_{i_{min}} < \sqrt{\left(cos(\frac{\theta}{2})xn_i - sin(\frac{\theta}{2})yn_i - \frac{v}{2}cos(\frac{\theta}{2} + \phi) - x_{cox}\right)^2 + \left(sin(\frac{\theta}{2})xn_i + cos(\frac{\theta}{2})yn_i - \frac{v}{2}sin(\frac{\theta}{2} + \phi) - y_{cox}\right)^2} < l_{i_{max}} \tag{18.12}$$

18.4.3 Solving for constrained parameters

The derivation of these six equations represents a total of six calculations, that needs to be taken made when passing a gait velocity v, an angle θ, and a rotation ϕ to the gait generation routine. If an invalid combination is passed, the constraints are broken, and the MR will either suffer from leg collisions, or try to access areas outside of the configuration space. To avoid these collisions, the AEP and PEP points should be moved in such a way that the constraints are not broken. As the position of the points are dependent on the three variables v,θ and ϕ the AEP and PEP points can be moved by changing any of these variables. To comply with the constraints one(or more) of these variables needs to be changed. It is clear that the angular constraints of equation 18.8 and 18.9 can simply be taken into account by truncatin any angle above θ_{max} to θ_{max} and any below θ_{min} to θ_{min}. By doing this, no effect will be seen on the gait, as these are physical constraints, that if broken will always result in the servo motor blocking. This entails that the set of constraints that needs to be taken into account have been reduced to 4.

These remaining constraints are both about describing the configuration space of the individual legs. As described, the AEP and PEP can be moved by changing one of the variables v,θ and ϕ. If an illegal AEP or PEP is passed, one or more of these variables should be changed, so that the constraint is obeyed. It is clear that if ϕ or θ are changed, the general movement direction of the MR is changed. As the direction of the MR is far more important than the

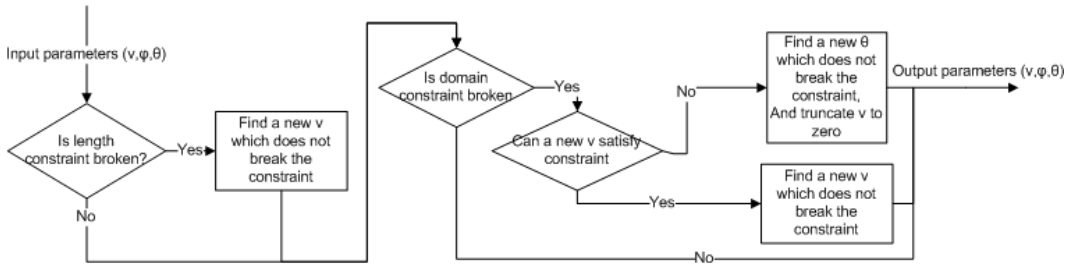

Figure 18.2: Shows a diagram of how the igiven AEP and PEP positions are changed, by changing the parameters of the gait, when constraints are present.

speed(See section 2), it is chosen to first decrease the speed indicator v, and then, if a feasible solution still does not exist, change the variable θ, so that a smaller rotation occurs. The procedure can be visualized from figure 18.2. The figure describe that new values for v and θ vales should be found in a way so that the imposed constraints are not broken. The solutions are found by means of maple, and can be seen in appendix C with a short description of the procedure. The derived equations are quite cumbersome, and to ensure readability of the thesis, the results a not repeated here.

As described earlier, the position of the AEP and PEP are also affected by a third constraint, the ZMP constraint. This constraint, and the trajectory between the AEP and PEP points are considered in the following chapter.

Chapter 19

Trajectory between AEP and PEP

Now that the AEP and PEP positions have been derived, it is time to investigate how the end effector should be moved, to obtain the best possible gait, given these endpoints. As described in section 18, the transition between AEP and PEP is affected by two constraints, which should be taken into account, these are:

- ZMP constraints
- Kinematic constraints

The First of these constraints, the ZMP constraint is perhaps the most important one to consider. It is this constraint that determines if the developed gait is stable, or if unwanted rotations around the polygon of stability will occur. The second one, the kinematic constraints determines if the end effectors of the MR slides upon the ground, or if all translations occur in the coxa joint, where it is supposed to. It is thus necessary to develop a trajectory which does not break the above constraints. In the requirements specification in section 3 a further demand was put on the gait generation process. It was specified that the generated gait should reduce the power consumption of the locomotion. It was also specified that a clearance of 2 cm were needed. To be able to obtain a gait which fulfills all of these requirements, it is suggested that the best approach will be to derive a performance function, and use numerical optimization techniques to obtain an optimal gait, given a set of weighting factors. This will be discussed in the following.

19.1 Performance function

As described in the previous section, it has been proposed to fulfill the large array of requirements by means of numerical optimization of a performance function. In the following this performance function is derived on a basis of these requirements.

19.1.1 Stability

The first part of the performance function that is included is the stability of the proposed gait. As described earlier, the stability of the gait is directly described by the ZMP point. If this point lies within the convex hull of the MR, the gait is dynamically stable. The farther away from the convex hull the ZMP is, the less likely the MR is to topple over. This entails that the performance measure of the MR when regarding stability is defined to be:

$$P_{stabillity} = k_1 \int |M_s(t)|dt \tag{19.1}$$

where:
$M_s(t)$ is a function dependent testing if the system is stable.

$$M_s(t) = \begin{cases} 0 \; if \; d_s > 0 \\ k \; if \; d_s \leq 0 \end{cases} \tag{19.2}$$

Where:
α is the weighting factor of the stability
d_s is the distance from ZMP to the convex hull
k, is a large constant added to the performace function if ZMP is outside of the convex hull, This constant is made 10.

The magnitude of K, should simply be big enough so that a solution which does not have a zmp outside of the complex hull is obtained.

19.1.2 Clearance

The third parameter of the performance function is the clearance of the end effector when in the return stroke phase. The clearance has two big impacts on the behaviour of the gait, the first one concerns the fact that the bigger the clearance, the less likely the MR is walk into obstacles when the legs are placed after the return stroke. The second impact is that the faster the MR removes it's legs from the ground during the return stroke, the faster the MR is able to move the body forward. As described in section 3, the legs should have a clearance

of at least 2 cm. This combined with the other characteristic results in the following penalty function

$$P_{Clearance} = k_2 \int \sum_{i=1}^{18} |f(t)|dt \tag{19.3}$$

where:
the function $f(t, z_i)$ is a function

$$f(t) = \begin{cases} 0 & if\ c_i < 2 \text{ and return phase} \\ (2 - c_i) & else\ 0 \end{cases} \tag{19.4}$$

where c_i is the clearance of leg i.

19.1.3 Joint torques

To minimize the power used by the servo motors, as stated in the requirements specification, the joint torque affecting the servos are also part of the optimization function.

$$P_{torques} = k_3 \int \sum_{i=1}^{18} |\tau_i(t)|dt \tag{19.5}$$

where: i is an index over all servo motors of the system
$\tau_i(t)$ denotes the load torque over servo i at time t

19.1.4 Displacement of centre of mass

It is of course also important that the gait is able to move the endpoints between the AEP and PEP in such a way that the MR actually moves. This entails that the travelled distance of the centre of mass is also included in the model. It is known that the MR should be able to move as far as described by the vector v, described in section 17.3, which entails that the following performance measure is used:

$$P_{Displacement} = k_4 \cdot \sqrt{v^2 - (P_{start} - P_{end})^2} \tag{19.6}$$

where:
P_{start} is the start position of the centre of mass of the mobile robot
P_{end} is the end position of the centre of mass of the mobile robot.

19.1.5 Impact forces

The next performance parameter that should be taken into accout is the impact forces of the MR. The smaller the impact forces, the smoother the gait. This will also generally reduce the power consumption of the gait, as the abrupt accelerations will result in used power in the servo motors. One might simply use a function, which uses the sum of squared end effector forces as the measure of how big the impact forces are, but as this would result in a result that is much more dependent on the values in steady state,than when the end effector strikes the ground hard, it has been chosen to use a function of the form:

$$P_{Impact} = k_5 \sum F_e^2 \tag{19.7}$$

Where F_e is the z component of the end effector force after it has been subject to a threshold.

$$F_e = \begin{cases} F_{e-z} \; if \; F_{e-z} > Th \\ 0 \quad if \; F_{e-z} \leq Th \end{cases} \tag{19.8}$$

where:
F_{e-z} is the z component of the end effector force
Th is the threshold of the function. This threshold is set to be the force acting on the end effectors of the MR, when the MR is standing on three legs in steady state. $Th = g \cdot m_{MR}/3 \approx 9$.

19.1.6 Tangential endpoint forces

The next performance parameter which should be included is the tangential endpoint forces. These forces describe how much friction the MR is subject to from the ground. A part of these tangential forces are the ones mowing the MR forward, but a part of these come from the kinematic constraints. The tangential forces are included in the following fashion:

$$P_{Tforces} = k_6 \cdot \sum (|F_{e-x}| + |F_{e-y}|) \tag{19.9}$$

where:
F_{e-x} is the x component of the end effector force
F_{e-y} is the y component of the end effector force

19.1.7 Smoothness

The final performance criteria is the smoothness of the given trajectory. To ensure that the given trajectory is not completely irregular, a smoothness criteria is imposed, which mathe-

matically can be derived as:

$$P_{smoothness} = k_7 \cdot \sum_{i=2}^{n} (Z_i - Z_{i-1}) \tag{19.10}$$

where:
Z_i is the z coordinate with index i
n is the number of cordinate references.

19.1.8 Collected performance function

Now that a set of parameters within which the gait should be optimal have been found, the complete optimization function can be seen to be:

$$\begin{aligned} P = \sum P &= P_{stabillity} + P_{Tforces} + P_{Clearance} + P_{torques} \\ &+ P_{Displacement} + P_{Impact} + P_{smoothness} \end{aligned} \tag{19.11}$$

19.2 Optimization domain

When describing a performance function, it is of course necessary to describe in what parameters the optimization should be done. As described earlier, it is the gait transactions between the AEP and PEP which should be optimized. This entails that the optimization procedure should provide a trajectory between AEP and PEP in x-y-z coordinates. However in [46, P3-4], where the foot trajectories of a cricket is analysed, the x-y trajectories from figure 19.1 has been found. From this figure, it can be seen that the movement of the hind legs can approximately be seen as linear both, when conducting the power stroke (stance phase) and the return stroke (Swing phase). The middle leg can be seen to have an approximately linear power stroke, but with a curve on the return stroke, And the front leg has a linear return stroke, and a curved power stroke. That is, during the power stroke, two of the legs have an approximately linear path between AEP and PEP. To simplify the optimization procedure, this linear approximation is extended to include the front legs, and to be valid both in the return and power stroke. This entails, that the optimization can be reduced to finding an optimal trajectory in the z axis direction. By chosing that the trajectory between AEP and PEP is a line, it is not necessary to check if the entire set of points defining the trajectory breaks the constraints defined in chapter 18, it is only necessary to check the endpoints.
This smaller optimization domain should however also provide a good way of reducing the impact forces, of the end effector, give a good clearance and reduce the torques affecting the

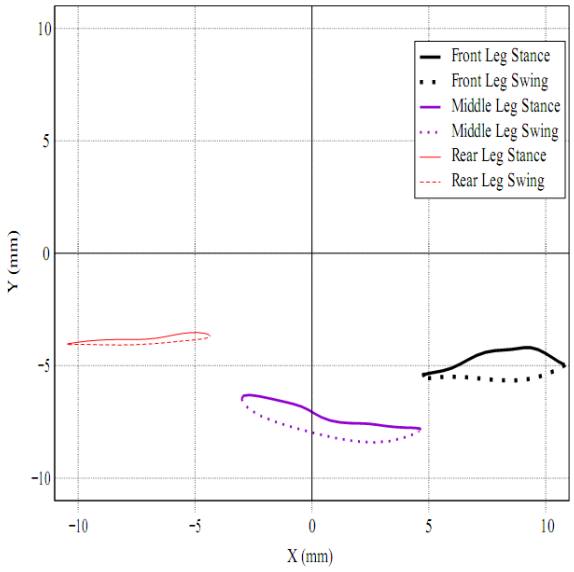

Figure 19.1: Shows the X-Y trajectories of of the legs of a cricket

servo motors. It can however prove insufficient in dealing with the tangential forces acting on the MR, which again is a direct indication that the real robotic system might begin to slide on the ground, which again would be a symptom of that the kinematic constraints have not been taken into account. The projection for this linear trajectory can be seen from figure 19.2, in the case where the MR moves forward, without turning.

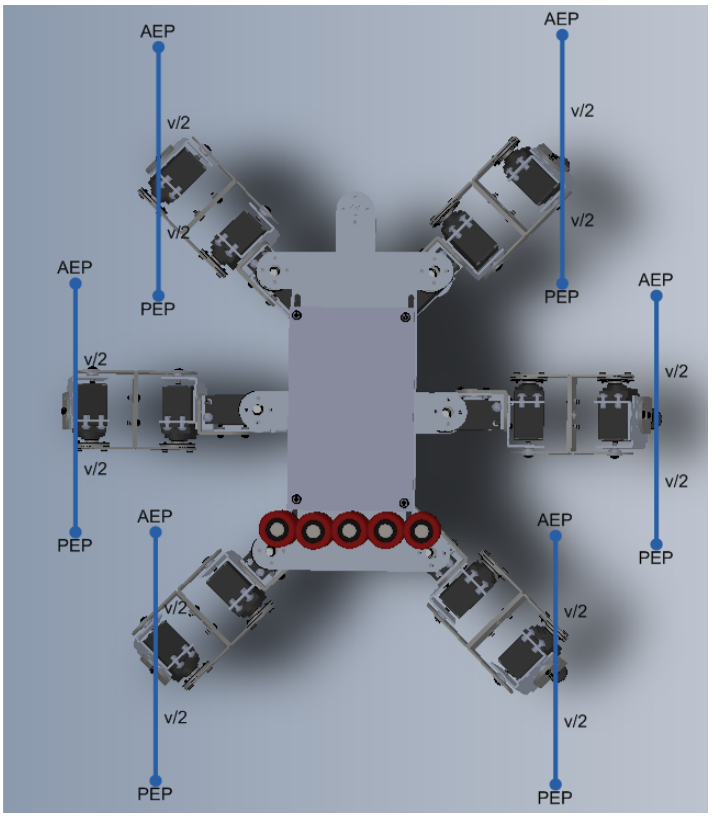

Figure 19.2: Shows the linear trajectory between the AEP and PEP points

19.3 Obtained solution

To obtain a solution for the optimization, the k values should be found. These are found by making a reference run of the simulation, and normalizing all obtained error variables to 1, with the exception of k_1, ehich is made equal to 1. The obtained constants were:

$$k_1 = 1 \tag{19.12}$$
$$k_2 = 3.1849e05 \tag{19.13}$$
$$k_3 = 8.7283e-5 \tag{19.14}$$
$$k_4 = 14.0056 \tag{19.15}$$
$$k_5 = 1.9289e-6 \tag{19.16}$$
$$k_6 = 1.3055e-6 \tag{19.17}$$
$$k_7 = 1.7857 \tag{19.18}$$
$$\tag{19.19}$$

The start guess of the trajectory in the z direction is a simple square function, which can be seen from figure 19.3. Where the z value during the power stroke is equal to -15cm, and during the return stroke -13 cm, as seen from the centre of the robot. When the performance

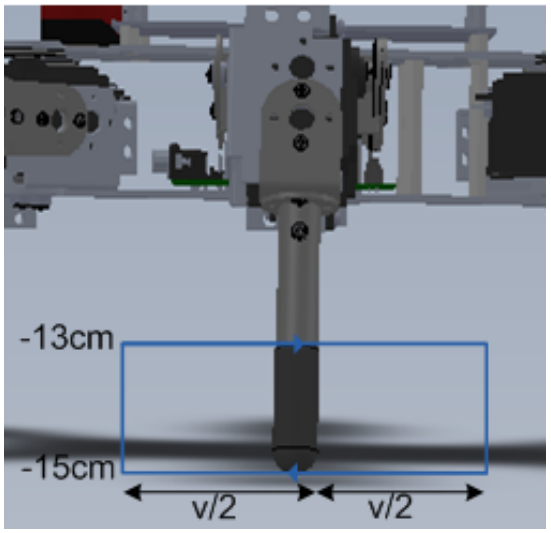

Figure 19.3: Shows the motion in the x-z plane, when the initial guess of the gait is used.

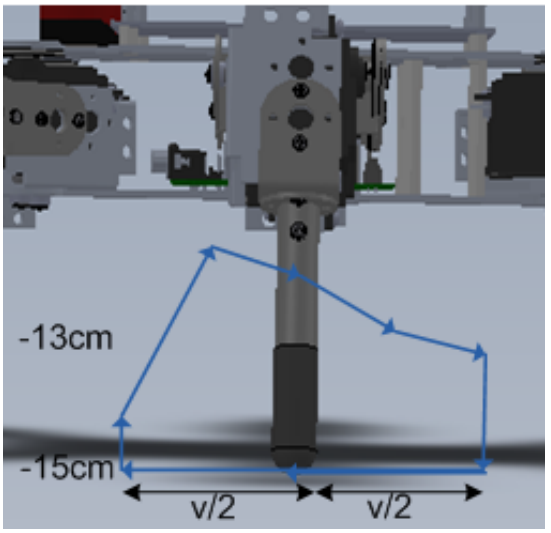

Figure 19.4: Shows the motion in the x-z plane, after the optimization has been completed

function described in section 19.1 is called using the matlab command fminsearch, with a space of 5 different values in the z axis the following 5 z axis values were found.

$$[-0.1402 - 0.1112 - 0.1161 - 0.1258 - 0.1301] \tag{19.20}$$

The total error of the system were found to be P=4.4724, distributed over the following parameters:

$$P_{Stabillity} = 0 \tag{19.21}$$
$$P_{Tforces} = 0.9163 \tag{19.22}$$
$$P_{Clearance} = 0.4905 \tag{19.23}$$
$$P_{Torques} = 0.9698 \tag{19.24}$$
$$P_{Displacement} = 1.0251 \tag{19.25}$$
$$P_{Impact} = 1.0120 \tag{19.26}$$
$$P_{Smoothness} = 0.0586 \tag{19.27}$$

As all were normalized to 1 in the beginning, it can be seen that the resulting gait has in fact optimized the torques and the tangential forces. It can however also be seen that the impact forces have increased. The most important result from this simulation is however that the stability parameter does not add anything to the collected error. This entails that the ZMP point is always inside the of the convex hull of the POS. It is clear that by defining a performance function, one simply shifts the tuning parameters of the gait from simply arbitrarily selecting trajectory points, to selecting the weighting factors of an optimization function. This entails, that the obtained trajectory can be formed simply by selecting appropriate weighting factors. One could compare the obtained optimal gait to the gait of sixlegged insects, which can be

said to have developed an optimal gait, to optimize it's chance of survival over the course of generations. Several studies of the gait, and walking patterns of insects have been made. In [46, P3-4] Figure 19.5 can be found, which describes the x,z foot trajectory of a cricket. From this figure, it can be seen that the derived optimal x-z trajectory in many ways are sim-

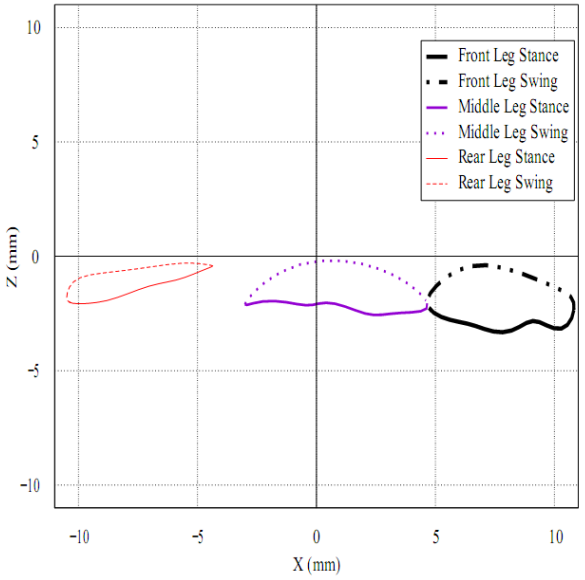

Figure 19.5: In [46, P3-4] an analysis of the foot trajectories of a cricket is conducted. This figure Shows the trajectory in the in the x-z plane

ilar, but generally have a more circular trajectory.

Now that the trajectory of between the AEP and PEP points have been defined, a complete gait has been characterized. It is now time to analyse how the gait can be used to control the hexapod robot.

Chapter 20

Controlling the hexapod

Now that a sufficiently good simulation model from chapter 7 has been obtained, a reference of how the line is situated in relation to the MR from chapter 15 and a way of determining how the MR should move by means of three simple parameters(θ, ϕ and v) from section 17, it is time to make a control algorithm for the MR. Generally the three parameters (θ, ϕ and v) should be able to describe the translation of the centre of the MR(v), the direction of this translation(ϕ) and the rotation per step (θ). In the following, the derivation of the controller structure will be given, and then a simulation is conducted. The capabilities of this controller will then be described in the acceptance test in section 21.

20.1 Controller structure

In the previous section it was mentioned that the inputs to the system are the three variables θ, ϕ and v. These three parameters should be used to make the MR able to track the line reference given by the computer vision system. As described in conclusion to part III the sensor data obtained consists of two points, one for each of the ends of the line. From these two points, a suitable control signal should be obtained. From Figure 20.1 the two reference points can be seen as the points P1 and P2, together with the position of the tip of the MR. The first thing that should be established is the error signal from which the control should be made. From Figure 20.1 it is seen, that the line, aligned with the x axis of the MR makes an angle Ω with the vector defined by the two points, P1(x1,y1) and P2(x2,y2). This angle could be used as a parameter for the control of the MR, as it would be sensible for the MR to always be parallel with the line. The drawback of this solution is however that the camera has a small viewing angle, and when turning to be parallel with the line, the camera would quickly lose track of the line position. This entails that it has been chosen to define the first

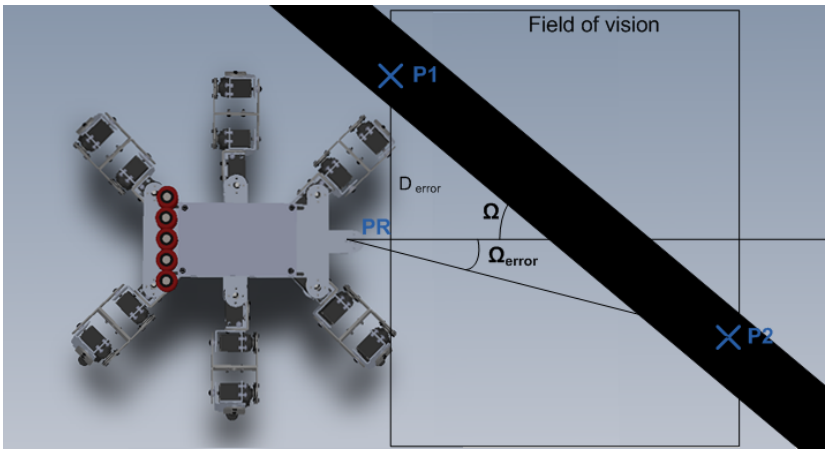

Figure 20.1: Shows the case, where the MR should track the line

error signal as the angular difference from the tip of the MR to P2, which will be labelled Ω_{error}. The second error signal should keep the MR on the line. It has been chosen to use the offset from the MR position, P1, as defined by D_{error} on Figure 20.1.

This entails that two error signals, and three control signals have been found. It was however defined in section 18, that the translational speed v should be most subject to change, when constraints are broken. This fact leads to, that it would be a poor choice to depend on this signal as a control signal, as it is frequently changed, especially, when large θ angles occur.

To simplify the control structure, two control loops are made, each with a PI controller. From Figure 20.2 this control structure can be seen.

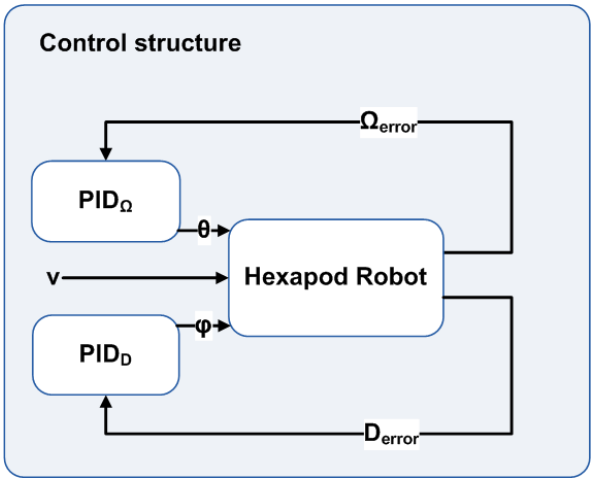

Figure 20.2: Shows the control structure of the MR. It can be seen that two PID controllers are used,

These two controllers are specified by two constants each:

$$\theta = C_{P\Omega} \cdot \Omega_{error} + C_{I\Omega} \int \Omega_{error} \tag{20.1}$$

$$\phi = C_{PD} \cdot D_{error} + C_{ID} \int D_{error} \tag{20.2}$$

As traditional frequency domain analysis does not apply to such unlinear systems, an analytical way of defining these parameters can not be established, and one must resort to an approach involving trial and error. From several simulations it has been concluded that a controller with the following parameters is a good choice.

$$C_{P\Omega} = -0.35 \tag{20.3}$$

$$C_{I\Omega} = -0.01 \tag{20.4}$$

$$C_{PD} = v16 \tag{20.5}$$

$$C_{ID} = 0.3 \tag{20.6}$$

20.2 Simulating the MR

Now that a controller structure has been established it is time, to test if the MR is able to track a line reference. Two simulations are shown in the following. One, where the MR is placed with an angle towards the line of 30 degrees, and with an offset of 10 cm, and the other where the MR, is placed parallel with the line, with a distance of 5 cm. These two simulations will be described in the following:

20.2.1 First simulation

The results of the first simulations regarding the position of the centre of the MR, and the rotation can be seen from Figure 20.3 and 20.4. From these figures, it is noted that the controller is able to move the MR close to the black line, with almost no steady state error. It is also seen that this happens with almost no overshoot. In the beginning of the simulation, the positional error increases. This is due to the fact that the legs of the MR in the beginning need to obtain the correct start configuration. The control of the gait first begins at time t=0.4s. Another interesting characteristic is that the MR is able to maintain a steady progression in the x axis direction, while correcting for the positional error. It is however also seen that small fluctuations occur, when considering the rotation of the MR. In Appendix G similar oscillations were seen, and it was concluded that they originate from the choice of friction model. This entails that it is concluded that these oscillations are caused by the

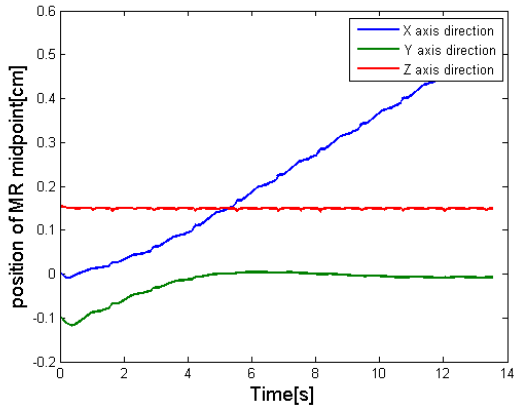

Figure 20.3: Shows the position of the centre of the robot

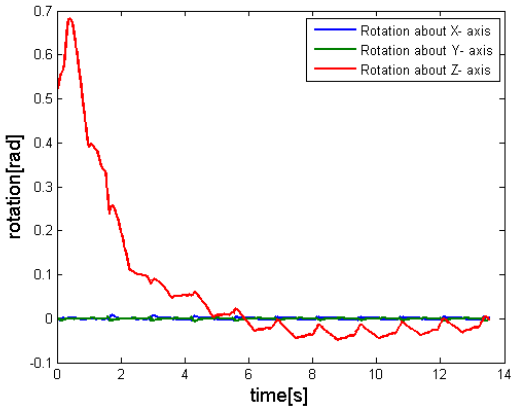

Figure 20.4: Shows the orientation of the mobile robot

system model, and not a marginally stable controller. The next thing that is considered is the control signal from the controller. These are seen from Figure 20.5 and 20.6.

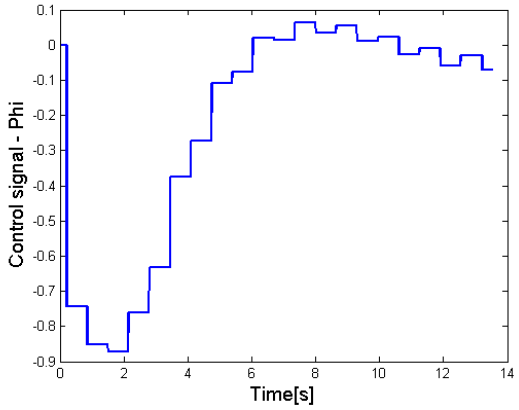

Figure 20.5: Shows the control signal, which defines the ϕ value of the gait

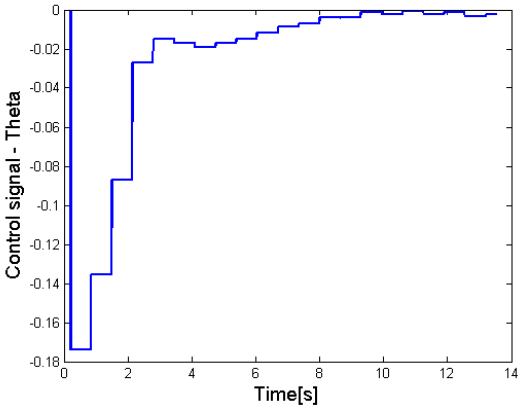

Figure 20.6: Shows the control signal, which defines the θ value of the gait

It is seen that these two control signals converge to zero, which means that the developed gait is reduced to walking in a straight line. To make it easier to visualise how the robot acts from the presented graphs, three plots of the robot pose can be seen from Figure 20.7 to 20.9. As described, it is vital to determine if the MR is stable, during the whole simulation. This entails, that the distance from the convex hull to the ZMP point should be considered. This distance can be seen from Figure 20.10 From this figure, it is seen that at times, the ZMP actually lies outside of the convex hull, as the distance is negative. This entails, that a stable gait cannot be guaranteed, even though the gait was stable, when the optimization was conducted in section 19. It is however also seen, that these negative distances always happen, while the MR is supported by all six legs, and that the simulation of the MR does not show any signs of the MR falling, or in any other way obtaining unwanted rotations. Further more,

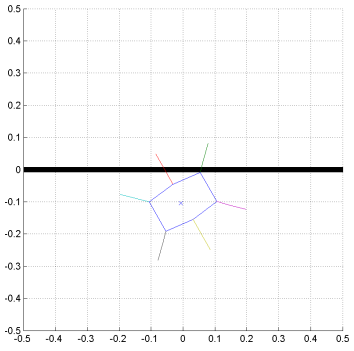

Figure 20.7: t=0s

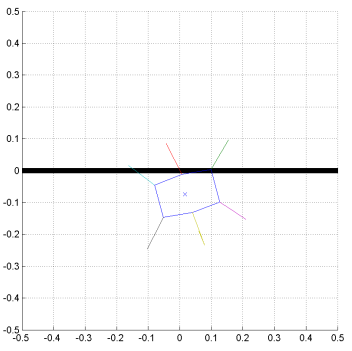

Figure 20.8: t=1.6s

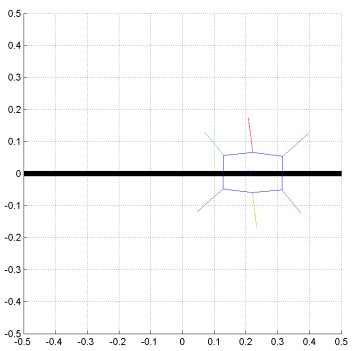

Figure 20.9: t=6.8s

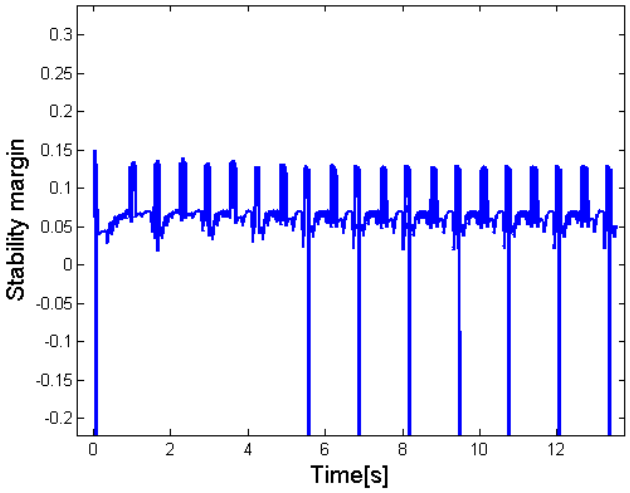

Figure 20.10: Shows the distance from the ZMP point to the convex hull

it also happens at the time, where the weight of the MR is shifted between the leg pairs. A close-up of one of these occurrences can be seen from Figure 20.11 with the corresponding leg forces shown on Figure 20.12.

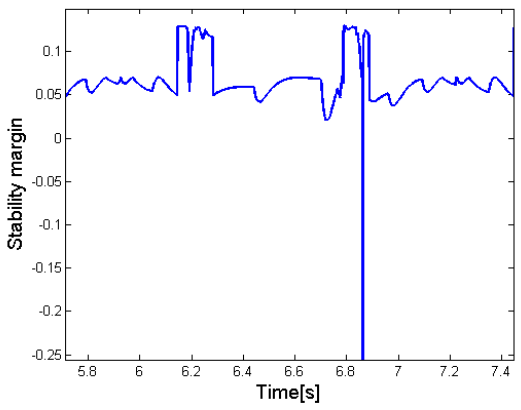

Figure 20.11: Shows a small section of Figure 20.10

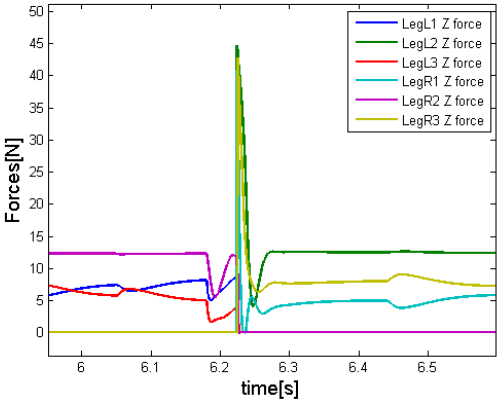

Figure 20.12: Shows the corresponding leg forces, acting at the time the ZMP distance crosses zero

From this figure, it is seen that very large changes in end point forces occur at these points in time. This might lead to big accelerations of the links of the MR, and thus to big changes in the ZMP of the MR. It is however noticed that these negative ZMP distances occur over a very limited timeframe. From section 9 it was described that if the ZMP is situated outside of the convex hull of the support polygon, accelerations leading to instability could occur. As the duration in which the ZMP is outside of the convex hull of the support polygon is very limited, it is assessed that the MR should be able to maintain a stable walk.

20.2.2 Second simulation

The second simulation is of the MR standing parallel with the line, in a distance of 5 cm. The results of the simulations regarding the position of the centre of the MR, and the rotation can be seen from Figure 20.13 and 20.14. From these plots the same tendencies from the previous

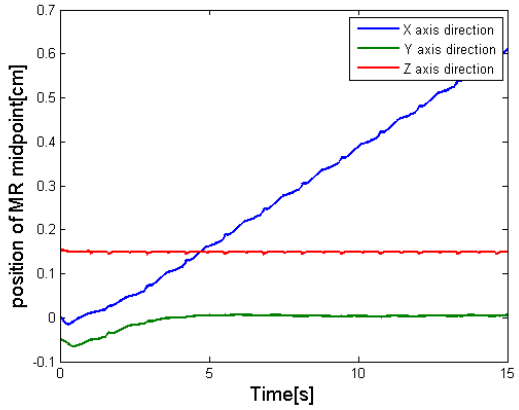

Figure 20.13: Shows the position of the centre of the robot

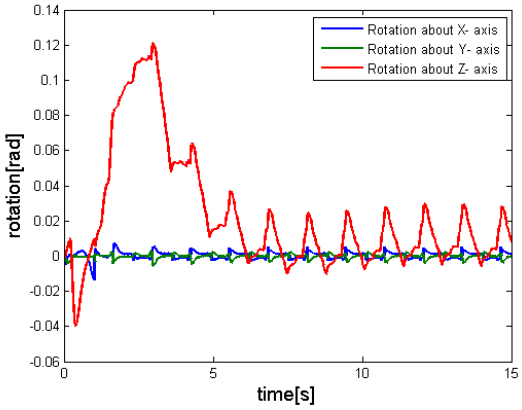

Figure 20.14: Shows the orientation of the mobile robot

simulation can be seen. The position converges towards the black line, while maintaining a steady velocity in the x direction. The same tendencies regarding oscillatory behaviour of the orientation can also be seen. The control signals can be seen from figure 20.15 and 20.16 From these two figures, it can again be concluded that the two control signals converge to zero, which means that the developed gait is reduced to walking in a straight line. To make the interpretation easier, plots of the pose of the MR can be seen from Figure 20.17 -20.19 Again it is important to check stability, and the distance from ZMP to the convex hull is calculated. The results can be seen from Figure 20.20.

It is seen that same tendencies regarding instability can be observed on this plot, and again this effect occurs when the MR changes which legs should carry the weight. As the pattern of these instabilities follow the same pattern as in the previous simulation. The same conclusion is made. That these indications of instability are of no concern when implementing the

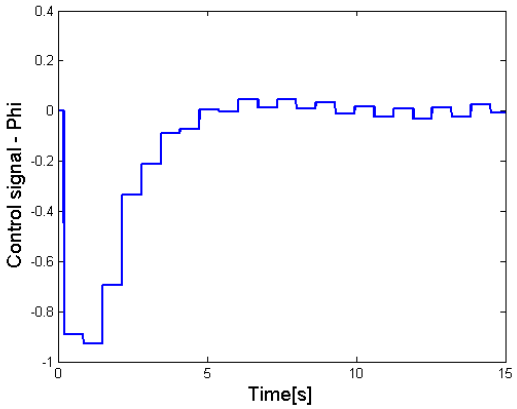

Figure 20.15: Shows the control signal, which defines the ϕ value of the gait

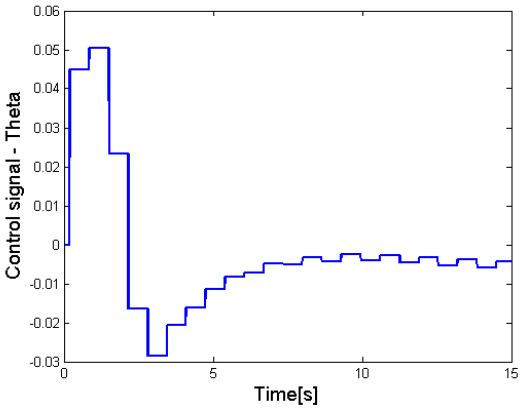

Figure 20.16: Shows the control signal, which defines the θ value of the gait

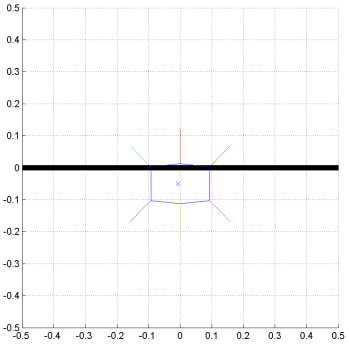

Figure 20.17: t=0s

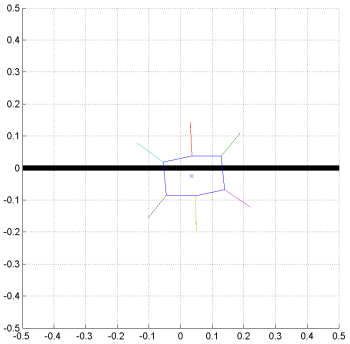

Figure 20.18: t=2.1s

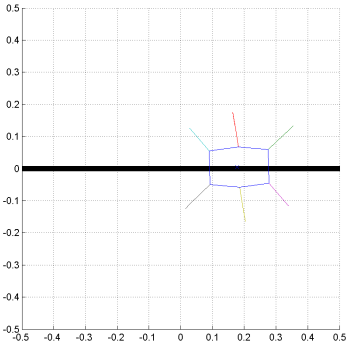

Figure 20.19: t=5.5s

control algorithm on the hexapod.

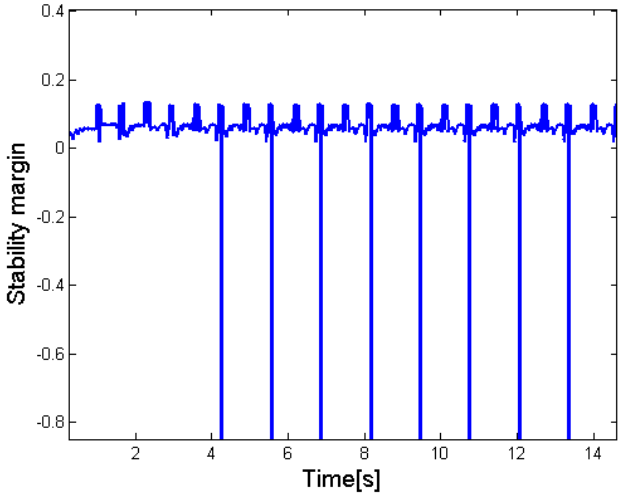

Figure 20.20: Shows the distance from the ZMP point to the convex hull

Part conclusion

In the previous section, a real world reference were derived from the image coordinate references obtained from the computer vision system. Then an analysis of how a gait should be developed were conducted, and the gait were specified by means of three parameters θ ϕ and v. Then an analysis of the optimal trajectory in the X-Z plane were made, and finally a controller based on two feed back loops were derived. This controller were simulated and concluded to be sufficiently good for further testing. In the next part of this thesis this controller is tested, and a conclusion regarding the system is made.

Part V

Conclusion and verification

Chapter 21

Acceptance test of integrated system

In this chapter, the requirements described in section 3 are tested. But, as described in the part conclusion of part III, the on board computer stopped responding during the development phase of the software system, which entailed that only a very simple implementation of the system was made in MATLAB. In this simple system, only the generated gait and the line tracking capabilities were implemented, which again entails that the requirements, regarding the software system, cannot be tested and verified. It is however concluded that the design of the system was made in such a way, that the requirements would have been fulfilled, given that the complete system could have been implemented.

21.1 Test of gait and robotic platform

The first series of tests are conducted to test the requirements to the physical and dynamic aspects of the mobile robot. In the requirements section the following requirements were presented:

- Line tracking
  The Mobile robot should be able to track the black line of the obstacle course.
- Dimensions
  The maximum dimensions of the MR should be

$$L \times W \times H = 40cm \times 43cm \times 25cm \tag{21.1}$$

- Clearance The MR should have a clearance of 8 cm from the ground to the bottom of the robot, and be able to climb obstacles 2 cm high.

- Speed
  The MR should be able to walk 25 cm/s, while tracking the line
- Move up/down a slope.
  The two slopes should enable the MR to move up a slope of 9 degrees, and down a slope of 14 degrees. Performance
  The performance of the gait should be taken into account when designing the gait generator

To conclude whether or not these requirements can be verified, a series of tests are conducted. The requirements to the dimensions of the MR are verified, as a new body is constructed.

21.1.1 Line tracking

In the requirements specification, it was described that the MR needs to be able to track a black line. To do this, a controller was developed in section 20. Several tests have been conducted, which can be seen on the CD in the folder "/tests/Controller". From these tests it is concluded that the MR performs quite good on both straight and curved line tracks. A thing that is not shown on these videos is the problem involved when the MR loses track of the line. This happens mostly when the black track is sloped, and is due to the small viewing angle of the camera. A video of this case is put in the folder "/tests/trackingerror".

21.1.2 Test of walking on a slope

In section 3 it was described that the MR should be able to walk up and down slopes of at least 14 degrees. To test this requirement a sheet of plywood were laid on the floor, with a black tape line attached. Then the level of one end of the sheet was increased, and the MR was instructed to move across the generated slope. From these tests is was concluded that the MR was able to move up/down the slopes, but the maximum "safe" incline was determined to be around 10 degrees. This entails that the requirement cannot be met. This was mainly due to the combination of two parameters. The first being that the face of the sheet of plywood was completely even and the legs of the robot had a tendency to slip upon the surface. On the webpage describing the obstacle course [12] it is however described that the steep incline is made rough to ease the transition. The second reason was the viewing angle of the vision system. If the MR was to slip, the black line quickly disappeared from its field of vision. It is however concluded that if the surface of the sheet of plywood was rough, the MR would be able to pass this requirement. A video of this test can be found on the CD attached to this report in the folder "/tests/10degreeslope.MTS".

21.1.3 Speed test

The next test determines if the MR is able to walk with the required minimum speed of 0.25 m/s. The test is conducted by laying out a track that is 3 meters long. The MR is started, and the time required to traverse the distance is recorded by means of a stop watch. The MR should reach the finish before 12 seconds have passed. From this test it is concluded that the MR is able to traverse a distance of 3 meters in a time of 12.66 s. This entails that the MR is not quite fast enough to be able to pass the first port on the RoboCup track, but it is very close. The video of this test can be found on the CD attached to this report in the folder "/tests/Speedtest.MTS".

21.1.4 Tests of clearance

As described in section 3 two requirements for clearance were given. The first one is concerning a clearance of at least 8 centimetres from the ground to the body. The clearance from the ground to the bottom of the robot was measured to be approximately 9.5 cm. This entails that this requirement is fulfilled.
The second clearance requirement involves the climbing of obstacles that are 2 cm high. A video of this is seen on the CD in the folder "tests/clear".

21.1.5 Performance of gait generator

In section 17, describing the gait generation process, an optimization of a reference gait was deviced. To test if this gait can be said to optimize the physical system, it is necessary to make a set of measurements. The only sensors, equipped on the MR, capable of detecting any changes in the gait characteristics are the pressure sensors. In section 4 these sensors were however concluded to be quite error prone and unreliable. To verify if the derived gait actually has optimized the locomotion of the MR, these sensors are however used to test the forces of the legs. The two gaits described in section 19 were implemented on the platform, and were programmed to take 5 steps, while the sensor data was recorded. If the gait has optimized the performance, the pressure on the end effectors, when they strike the ground, should decrease. The results from this test can be seen from figure 21.1 and a small section can be seen from figure 21.2. This test was conducted for all six legs. Two of these tests ended up with the error described in section 4, where the rubber shielding created an offset of the ground force. From the above figures it is seen that there is little difference between the optimized gait and the reference gait, when considering the steady state values. This is as expected, as the legs still need to carry the weight of the MR when they are on the ground.

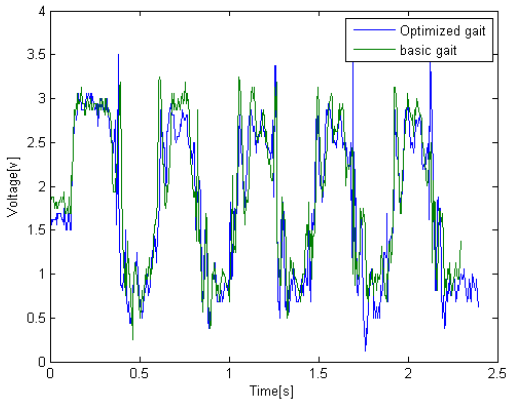

Figure 21.1: Shows the voltage over the pressure sensor, when the five steps of the two various gaits are applied

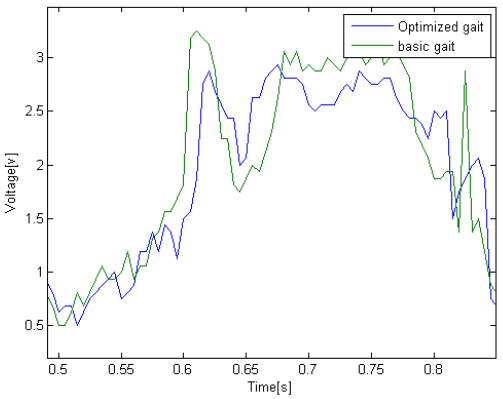

Figure 21.2: Shows a section of figure 21.1, from which it can be seen that no big difference in force magnitude is observed.

The difference should be when the leg strikes the ground. From figure 21.2 a small section of graph 21.1 is seen. From this, it is seen that the basic gait has a tendency to climb a bit faster, and a bit more than the optimized one. This effect is however so small that no definite conclusion can be made.

21.1.6 Summary

From the performed tests, it was concluded that the MR is able to track the line in most circumstances. Sometimes it does however have problems when moving on curved paths, as the viewing angle of the camera is very small. It was also concluded that the MR was able to climb slopes of approximately 10 degrees, and objects two centimetres high. It was however also seen that the MR was unable to make the speed requirement, by a margin of 0.66 seconds. The final conclusion of this set of tests was, that no conclusive change between the derived optimal gait, and the reference gait could be seen from measurements.

The next set of tests, that should be made, are the tests of the vision system.

21.2 Test of vision system

In the requirements specification, several requirements were made regarding sensor input and its processing.

- Discern black, grey and white areas

To find its way around the track, the MR needs to be able to track a black line over a white floor, and even on grey tiles.

- Robustness
  The MR should be able to produce results regarding where to move on the line, even though the path curves, or other objects, are placed close to the track.

- The line tracking software
  The line tracking algorithm should provide a basis on which object recognition could be built.

21.2.1 Test of background colour

In the following test it is examined if the implemented computer vision system is able to track the line, and provide a good ground reference, even when the background colour changes. In section 12 it was tested on a grey background. And now it is tested on one white (composed of sheets of paper), and one yellow/brown (composed of a sheet of plywood). As the computer vision system works on contrasts, and not colour/intensity, it should be invariant to colour changes. The intensity information from the red channel of the camera, the straight line representation, and the goal locations are presented on figure 21.3 to 21.6. From these

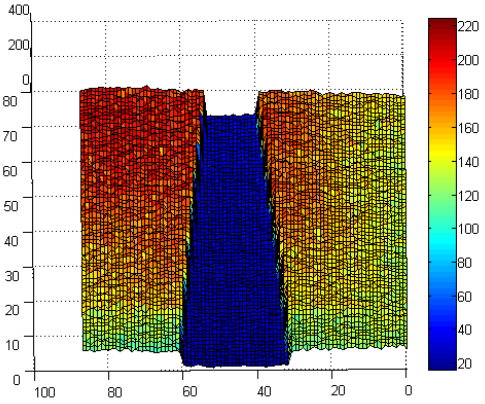

Figure 21.3: Shows the intensity of the red channel, when an image is taken of the black line on a white background

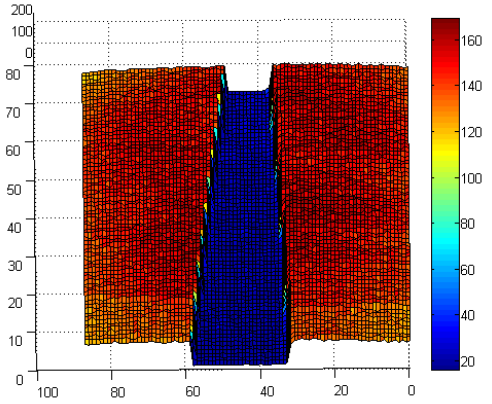

Figure 21.4: Shows the intensity of the red channel, when an image is taken of the black line on a brown/yellow background

tests it can be seen that the computer vision system, in both cases, is able to produce a goal location. It is however also seen that the main intensity of the background changes, due to the darker surface of the sheet of plywood. Another thing that is noticed from the figure is the fact that each side of the line is only represented by a single linear approximation. From these results it is concluded that the vision system works both on a white, grey and

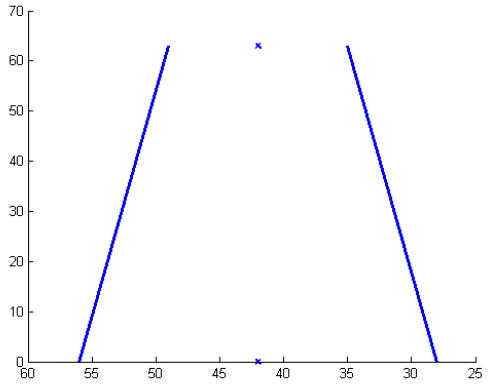

Figure 21.5: Shows the goal locations and the straight line representation of the black line on a white background

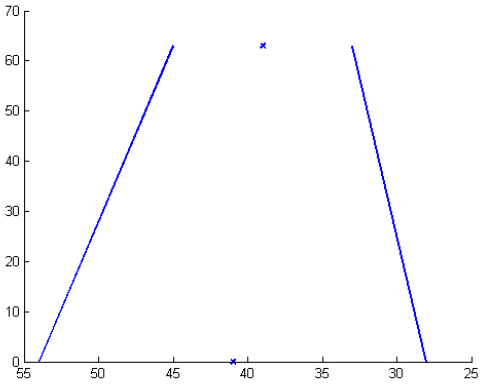

Figure 21.6: Shows the goal locations and the straight line representation of the black line on a brown/yellow background

brown background. The test results can be found on the CD in the folder "tests/vision/" in the subfolders "brownbackground" and "whitebackground".

21.2.2 Robustness

In section 12 it was concluded that the computer vision system was robust, when regarding objects close to the track. To further test this statement, tests on curved paths are conducted. To see if the vision system works on curved paths a small test is made, where the MR is placed close to a slope. An image is saved, and the locations of the goals from the image are identified. The results of this test can be seen from figure 21.7 and 21.8.

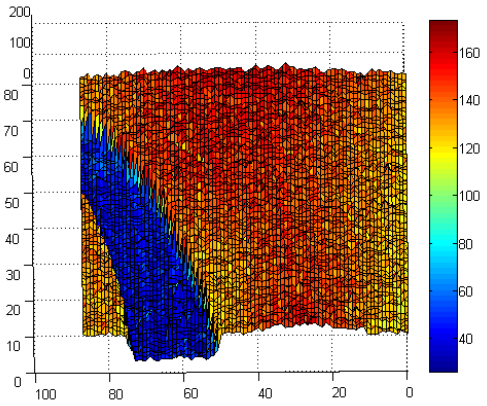

Figure 21.7: Shows the intensity of the red channel, when an image is taken of a curved black line.

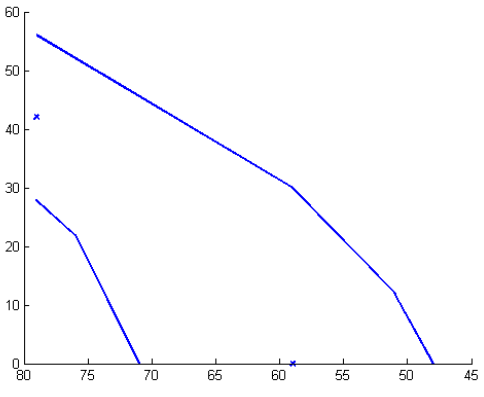

Figure 21.8: Shows the goal locations and the straight line representation of a curved black line

From these images, it is seen that the algorithm is able to produce goal locations when re-

garding a curved path. It is seen that the right side of the curved path is composed of three straight line segments and the left of two. Four independent tests were made, all of which are included on the CD (in the folders "curve1" to "curve4"). The results presented above are from the fourth dataset.

21.2.3 Summary

Two tests were conducted to verify the computer vision system. One to verify that the detection of the line positions were invariant of background colour, and one to further test the robustness of the implemented algorithm. Both of these tests were successful, and it is concluded that the computer vision system works as intended.

Chapter 22

Conclusion

In the following chapter, a conclusion on the results, obtained through the course of this master thesis, will be given. As many results have been obtained, the conclusion will first sum up the main results from the various parts of the report, and then a conclusion in more general matters is made at the end. Generally the thesis was divided into four main parts, a preliminary analysis, a part concerning the mathematical model of the system, a part concerning the software system, and finally one concerning the various aspects of controlling the robot.

Preliminary analysis

In the preliminary analysis, it was described that legged mobile robots had several advantages over robots with locomotion based on wheels. This entailed that it was proposed to construct an autonomous mobile robot capable of performing tasks in a given environment. It was then concluded that the track from a competition for mobile robots would serve as a good way of testing the capabilities of the developed robotic system. From the environment of the RoboCup track, a series of requirements were proposed, and from these requirements, a platform for the mobile robot was constructed.

Modelling

The second part of the thesis concerned the mathematical model of the system. In this section, three distinct models were developed. First the kinematic model, capable of describing the position of the parts of the robot by means of the angles of the legs of the robotic system. The next model that was derived was the inverse kinematic model of the system. This model

solved the completely opposite problem - the problem of deriving the angles of the legs by means of the positions of the end points of the legs. The third model that was derived was the dynamic system model, which described the forces acting on the legs of the robot. Six of these models, one for each leg, were combined with the dynamic equations of a free body, to yield a complete dynamic model of the mobile robot. The two kinematic models were verified, one visually, and one by testing on the physical system. The dynamic model could however not be verified due to the lack of sensors providing information regarding the motion of the robot. Instead the model was concluded to simulate a "six legged walking machine". It was however concluded that the model would have benefitted from a better friction model.

Software and intelligence

In the part of the thesis concerning the software system, a design of the software of the mobile robot was made. The analysis of how the MR should navigate in the environment was decomposed into four sub problems. The problems of perception, localization, cognition, and motion control. To provide the MR with vision capabilities, a computer vision system was developed and tested. The problem of localization was only graced, as it had been chosen not to implement a localization algorithm. In the section regarding cognition, a complete system was designed, which enabled the MR to move in the environment, by means of a path planning algorithm. This system never became in functional order, as the microprocessor board, on which it should have been implemented, stopped responding. To perform the acceptance test, a small backup system was implemented, which interfaced the hardware components via MATLAB.

Control

In the final part of the thesis, the concept of how the MR should attain locomotion was addressed. The first thing that needed to be done was to derive a real world control reference from the image reference found by the computer vision system. Then a small discussion followed, where it was defined that a simple description of a tripod gait was needed. This lead to a description, which defined the tripod gait by means of three parameters. Afterwards the constraints of the leg positions were considered, and finally a gait trajectory was developed by using numerical optimization techniques. The final chapter of this part concerned the development of a controller, which should provide the MR with line tracking capabilities. In the acceptance test it was concluded that the derived controller was able to make the MR track the line.

From above it is seen that many good results have been obtained while working on this master thesis. The biggest drawback when regarding the robotic platform was the lack of installed sensors. This lead to the consequence that no definitive conclusion regarding the dynamic model of the system could be reached, and that the derived optimal gait could not be properly tested. Generally it is hard to make any definitive conclusions regarding how the proposed system would perform, if it was to participate in the competition. This is mainly due to the fact that hardware issues made the implementation of the software system impossible. It should however be noted that despite these difficulties, a working mobile robot has been constructed, with a computer vision system capable of providing the MR information about the surroundings, and with a controller making it capable of line tracking.

Bibliography

[1] Roland Siegwart and Illah R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. The MIT press, 2004.

[2] Gregory Dudek and Michael Jenkin, *Computational principles of mobile robotics*. Cambridge university press, 2000.

[3] Freyr Hardarson, "Locomotion for difficult terrain-A Survey Study," 1997.

[4] R. h. MIT. `http://www.ai.mit.edu/projects/leglab/robots/3D_hopper/3D_hopper.html`.

[5] A. Honda. `http://world.honda.com/ASIMO/`.

[6] "Boston dynamics, big dog." `http://www.bostondynamics.com/robot_bigdog.html`.

[7] "Boston dynamics, rhex." `http://www.bostondynamics.com/robot_rhex.html`.

[8] Mohiuddin Ahmed, Md. Raisuddin Khan, Md. Masum Billah and Soheli Farhana, "A Novel Navigation Algorithm for Hexagonal Hexapod Robot ," *American J. of Engineering and Applied Sciences 3 (2)*, vol. 229, pp. 320–327, 2010.

[9] H. Ohroku,K. Nonami, "Omni-directional Vision and 3D Animation Based Teleoperation of Hydraulically Actuated Hexapod Robot COMET-IV ," *ICGST-ARAS Journal*, vol. 09, 2009.

[10] Mohiuddin Ahmed, Md. Raisuddin Khan, Md. Masum Billah and Soheli Farhana , "Walking Hexapod Robot in Disaster Recovery: Developing Algorithm for Terrain Negotiation and Navigation,"

[11] Mads Jensen, Rasmus Pedersen, Jeppe Juel Petersen, Casper Lyngesen, Mogensen, Henrik Dalsager Christensen, "Modelling and Control of a Hexapod Robot," tech. rep., Aalborg university, 6 2009.

[12] T. university of denmark, "Description of Robocup track." http://www.dtu.dk/subsites/robocup/English/competition/track.aspx.

[13] L. motion. http://www.lynxmotion.com/.

[14] John J. Craig, *Introduction to robotics, Mechanics and control*. Addison-Wesley publishing company, 2003.

[15] Alan P. Bowling, "Mass Distribution Effects on Dynamic Performance of a Cable-Driven Hexapod,"

[16] ROGER D. QUINN and ROY E. RITZMANN, "Construction of a Hexapod Robot with Cockroach Kinematics Benefits both Robotics and Biology," *Connection Science*, vol. Vol. 10, pp. p239–254, 1998.

[17] Hitec, "ANNOUNCED SPECIFICATION OF HS-645MG STANDARD DELUXE HIGH TORQUE SERVO," 2005.

[18] Lynx Motion, "SSC-32 Ver 2.0-Manual written for firmware version SSC32-1.06XE," tech. rep., Lynx Motion, 2005.

[19] Interlink electronics, "FSR® Integration Guide and Evaluation Parts CatalogWith Suggested Electrical Interfaces," tech. rep., Interlink electronics.

[20] CMUcam3, "CMUcam3 Datasheet-Embedded Vision Processor," tech. rep., Carnegie Mellon University, 2007.

[21] Technologic systems, "TS-7400 DATASHEET," tech. rep., Technologic systems, Jun. 2009.

[22] JENS CHRISTENSEN - JESPER LUNDGAARD NIELSEN MADS SØLVER SVENDSEN - PETER FALKESGAARD ØRTS, "Development, Modeling And Control of A Humanoid Robot," 2007.

[23] Technologic systems, "TS-7400 Manual," tech. rep., Technologic systems, May. 2010.

[24] T. systems, "TS-7400 repository." ftp://ftp.embeddedarm.com/ts-arm-sbc/ts-7400-linux/.

[25] "CMUcam3 virtual cam wiki." http://www.cmucam.org/wiki/virtual-cam.

[26] R. Manseur, "Denavit-Hartenberg Parameters ." http://uwf.edu/ria/robotics/robotdraw/DH$_$parm.htm.

[27] David Baraff, *An introduction to physically based modelling*. Robotics Institute, Carnegie Mellon University, 1997.

[28] David Baraff, "Fast Contact Force Computation for Nonpenetrating Rigid Bodies," *COMPUTER GRAPHICS Proceedings*, vol. Annual Conference Series,1994, pp. "23–34", July 1994.

[29] Yariv Bachar, "Developing Controllers for Biped Humanoid Locomotion," 2004.

[30] Miomir Vukobratic, Branislav Boravac, "Zero Moment Point - Thirty Five Years Of Its Life," *International Journal of Humanoid Robotics*, vol. 1 NO. 1, pp. 157–173, April 2004.

[31] T.Arakawa and T. Fukuda, "Natural motion of biped locomotion robot using hierarchial trajectory generation method consisting of GA, EP, layers,," *International Conference on Robotics and Automation*, pp. 211–216, 1997.

[32] R. M. SHAPLEY AND D. J. TOLHURST, "EDGE DETECTORS IN HUMAN VISION," *Journal of Physiology*, vol. 229, pp. 165–183, 1973.

[33] Mike Heath,Sudeep Sarkar,Thomas Sanocki and Kevin Bowyery, "Comparison of Edge Detectors," *COMPUTER VISION AND IMAGE UNDERSTANDING*, vol. Vol. 69 No. 1, pp. 38–54, 1998.

[34] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[35] M. Andersen, R. Andersen, C. Larsen, T. B. Moeslund, and O. Madsen, "Interactive assembly guide using augmented reality," in *ISVC '09: Proceedings of the 5th International Symposium on Advances in Visual Computing*, (Berlin, Heidelberg), pp. 999–1008, Springer-Verlag, 2009.

[36] g. m. bobdavies2000, bornet, "Open computer vision library." http://sourceforge.net/projects/opencvlibrary/.

[37] Richard O. Duda,Peter E. Hart, David G. Storck, *Pattern Classification*. Wiley Interscience, 2001.

[38] Christian Nilsson, "Heuristics for the Traveling Salesman Problem," -, vol. -, pp. –, -.

[39] W. L. Pearn and T. C. Wu, "ALGORITHMS FOR THE RURAL POSTMAN PROBLEM ," *Computers Ops Res*, vol. 22, no. 8, 1995.

[40] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, 1997.

[41] I. P. Marija Seder, Kristijan Macek, "An integrated approach to real-time mobile robot control in partially known indoor environments," in *Industrial Electronics Society. 31st Annual Conference of IEEE*, 2005.

[42] Marija Seder and Ivan Petrovi´c, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," *Robotics and Automation, 2007 IEEE International Conference on* , pp. p 1986–1991, 2007.

[43] P. Lester, "A* algorithm." `http://www.policyalmanac.org/games/aStarTutorial.htm`.

[44] T. university of denmark, "Robocup." http://www.dtu.dk/subsites/robocup.aspx.

[45] C. Ferrell, "Robust and adaptive locomotion of an autonomous hexapod," in *From Perception to Action Conference, 1994., Proceedings*, pp. 66 – 77, sept. 1994.

[46] R. D. Q. Sathaporn Laksanacharoen and R. E. Ritzmann, "Modeling of insect's legs by inverse kinematics analysis," in *Proceedings of the 2nd international symposium on adaptive motion of animals and machines*, 2003.

Part VI

Appendix

Appendix A

Transformation matrices of the kinematic model

In the following section the various rotation matrices used in the derivation of the direct and inverse kinemtaic model is used.

A.1 Transformation from Robot frame to leg frames

The first six that are presented are the transformations from the central coordinate frame to the six leg frames. These are all different, but constant valued as there are no axes of rotation. They are all formed by the common notation:

$$\mathcal{T} = \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 & X \\ sin(\phi) & cos(\phi) & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.1}$$

Where the values for the offset and angle can be found from table A.1.

$$\overset{P}{_{L1}}\mathcal{T} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 9.25 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 5.25 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \overset{P}{_{L2}}\mathcal{T} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 6.25 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

Leg	Name	x[cm]	y[cm]	ϕ[Degrees]
Left 1	${}^{P}_{L1}\mathcal{T}$	9.25	5.25	45
Left 2	${}^{P}_{L2}\mathcal{T}$	0	6.25	90
Left 3	${}^{P}_{L3}\mathcal{T}$	-9.25	5.25	135
Right 1	${}^{P}_{R1}\mathcal{T}$	9.25	-5.25	-45
Right 2	${}^{P}_{R2}\mathcal{T}$	0	6.25	-90
Right 3	${}^{P}_{R3}\mathcal{T}$	-9.25	-5.25	-135

Table A.1: Shows the naming and parameters of the derived transformation matrices

$$ {}^{P}_{L3}\mathcal{T} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & -9.25 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 5.25 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}_{R1}\mathcal{T} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 9.25 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & -5.25 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.3} $$

$$ {}^{P}_{R2}\mathcal{T} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -6.25 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}_{R3}\mathcal{T} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & -9.25 \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & -5.25 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4} $$

Appendix B

Mechanical composition of hexapod robot

In the following section, the mechanical composition of the hexapod robot is discussed.

B.1 General apperance

Generally the hexapod robot is based upon the AH3-R Walking Robot, which is a hexapod robot manufactured by Lynxmotion [13]. The original robot can be seen from figure B.1 In

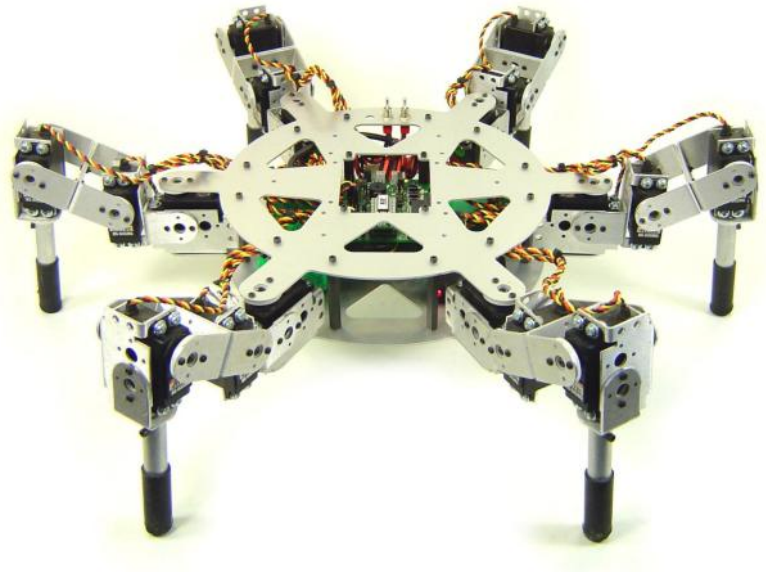

Figure B.1: Shows the hexapod robot with the original body from the manufactorers

the original configuration as shown from the figure, the robot is 45 cm wide [13] as the gates of the obstacle course are only 42 cm wide, a body modification is needed to make the robot

smaller. Another thing that should be made is a place to store the electronics used in the project. This results in a smaller body with a small rack mounted on the back of the robot for electronics. In the next section, which describes the making of a 3d model with similar physical parameters, an image of the final robot design is presented.

B.2 Making the 3d model

When determining the forces and tourques acting on the robot when the robot moves its legs it is necessary to know exactly how the weight distribution of the moving parts are. This weight distribution can be described by two factors, the centre of mass and the inertia tensor. The centre of mass describes the mean location of all the masses of the system, and the inertia tensor describes the distribution of these masses. These quantities can be hard to find by means of imperical measures or applying first principle to the physical object. Another way to obtain these is to make an accurate 3d model of the robot. As a lot of cad parts of the robot are directly available from lynxmotions website the cad software SolidWorks have been used to make an accurate cad model of the robot. The end result can be seen from Figure B.2.

In this model, all alluminium parts, standoffs, screws, nuts, washers, servo horns and rubber coatings have been found from lynx motions homepage. The body and rack have been modelled and the servo motor and batteries have been found on a cad drawing library on the internet. The solid works parts used for generating the model can be found on the CD in the folder CADMODEL, where the main assembly file is called aranya.SLDASM. The most important part of this model is that the weights are correct, as they effect the forces and torques when the servo motors are moved. To test if these weights are correct, several of the subparts of the model have been weighted and compared to the parts on the real model. In table B.1 these weights are shown

From these tests it is seen that the weights of the cad model are quite similar to the weights of the physical MR. To make the model as accurate as possible, extra weight were distributed on the surfaces of the some of the parts, to include the extra electronics, and wires. This was bone by increasing the density of the metal. This effect is most severe on the rack plates, as the electronics on these have not been included. As the electronics are spread evenly over the rack plates, this increase in density is assumed to be a good approximation. It is also noted that the servo weight matches the weight of the cad servo, although the cad servo is not modelled with wires. This entails that the person which have modelled the servo, have included the weight of the wire somewhere in the servo case. This will change the centre of mass of the servo, but as the wire weights significantly less than the rest of the

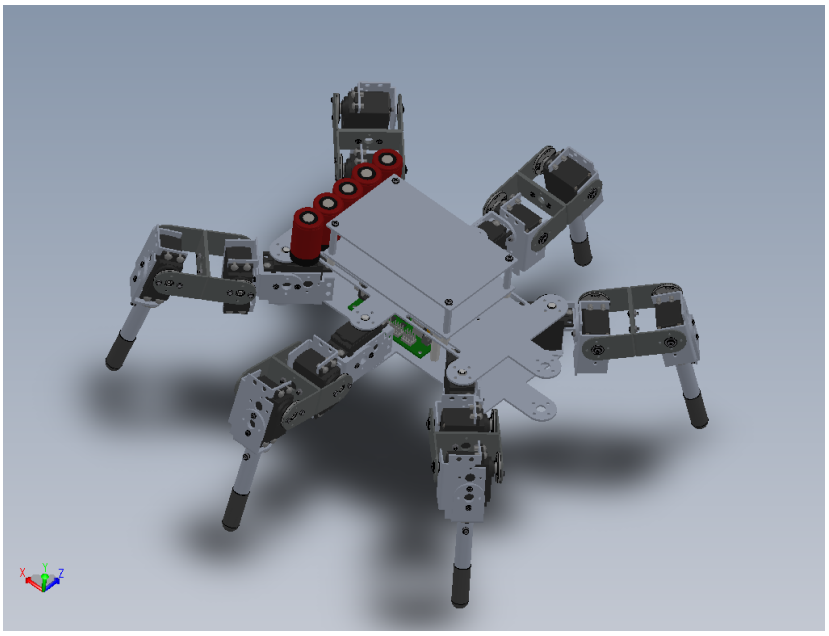

Figure B.2: Shows the constructed 3d cad model of the Lynxmotion robot with a new body

servo, the model is used, and assumed to be good enough. As none of the weights deviate significantly(all under a 5% deviation) from the measured weights, the individual parts, and the entire model is assumed to be correct, when extracting the centre of masses, and inertia tensors from SolidWorks.

Name of component	Measured weight[g]	Simulated weight[g]	Difference[%]
battery	273	262	-4.0293
servo motor(With wire)	57	56.97	-0.0526
rack	249	248.96	-0.0161
top	120.6	120.3	-0.2488
bottom	144.7	144.34	-0.2488
ssc-32	48.1	49.37	2.6403
Coxa link 1 and 2	156.4	158.1	1.0870
Femur link	34.1	34.1	0
Tibia link	102.5	102.2	-0.2927
total	2608	2596.88	-0.4264

Table B.1: Shows the naming and parameters of the derived transformation matrices

B.3 Weight distribution

The main virtue of the derived model is to provide information regarding the weight distribution of the robot. In the previous section a weighting of the MR was conducted, and compared to the weights of the constructed 3d cad model. In the following section, the inertia tensors and centre of mass of the various parts of the MR have been found.

Note that the CMUcam3 camera has not been included in the cad model, as it was attached to the MR very late in the writing of this thesis. This entails that the found parameters for the body, on which the camera is attached will be subject to an error, but the parameters for the legs should be correct

As the MR is composed of 19 moving parts, three for each leg, and one for the body, it is necessary to extract the parameters for all 19. However one must recognise that all femur and tibia links are alike, and that there only exists 2 different coxa links. This reduces the number of different parts to a total of five(1(Tibia)+1(Femur)+2(Coxa)+body=5) different parts. The parameters for these are shown on figure B.3-B.6.

One could verify each of the 5 inertia tensors and centre of masses by measurements. This Equipment is however not available at the department, and thus the parameters can not be verified. This entails that no definitive conclusion regarding wheter or not they are correct can be drawn. But as the underlying model, from which these parameters have been extracted has been verified, they are considered as being correct.

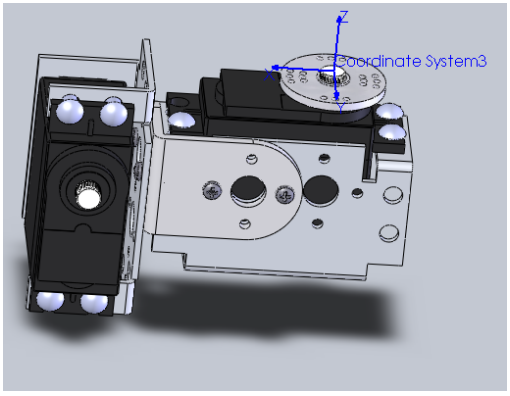

Figure B.3: Shows the coxa link for the four front and rear legs

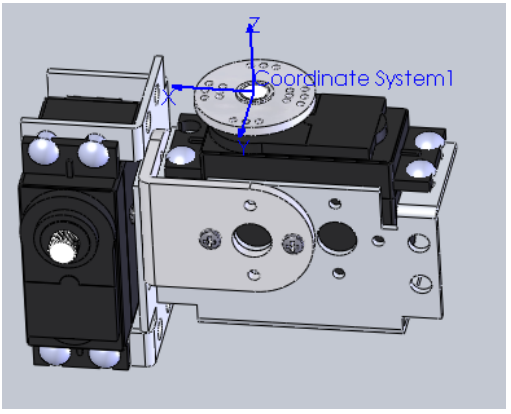

Figure B.4: Shows the coxa link for the two middle legs

$$IC = \begin{bmatrix} 0.04566 & 0.00283 & -0.01788 \\ 0.00283 & 0.12329 & -0.00128 \\ -0.01788 & -0.00128 & 0.10849 \end{bmatrix} \cdot 10^{-3}$$

$$Pc = \begin{bmatrix} 0.03123962 \\ 0.00262974 \\ -0.02684437 \end{bmatrix}$$

$$m = 0.15848800$$

$$Ic = \begin{bmatrix} 0.04666 & 0.00249 & -0.01821 \\ 0.00249 & 0.12083 & -0.00138 \\ -0.01821 & -0.00138 & 0.10497 \end{bmatrix} \cdot 10^{-3}$$

$$Pc = \begin{bmatrix} 0.01221700 \\ 0.00261163 \\ -0.02742362 \end{bmatrix}$$

$$m = 0.15848800$$

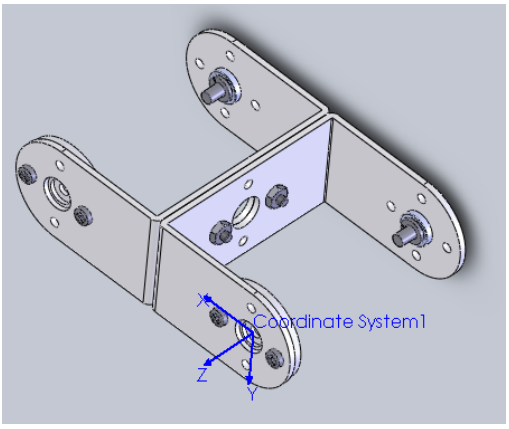

Figure B.5: Shows the coxa link for the four front and rear legs

$$
\begin{aligned}
Ic &= \begin{bmatrix} 0.00001777 & 0.00000000 & 0.00000000 \\ 0.00000000 & 0.00002856 & 0.00000000 \\ 0.00000000 & 0.00000000 & 0.00001380 \end{bmatrix} \\
Pc &= \begin{bmatrix} 0.02854148 \\ 0.00000000 \\ -0.02408866 \end{bmatrix} \\
m &= 0.03411514
\end{aligned}
$$

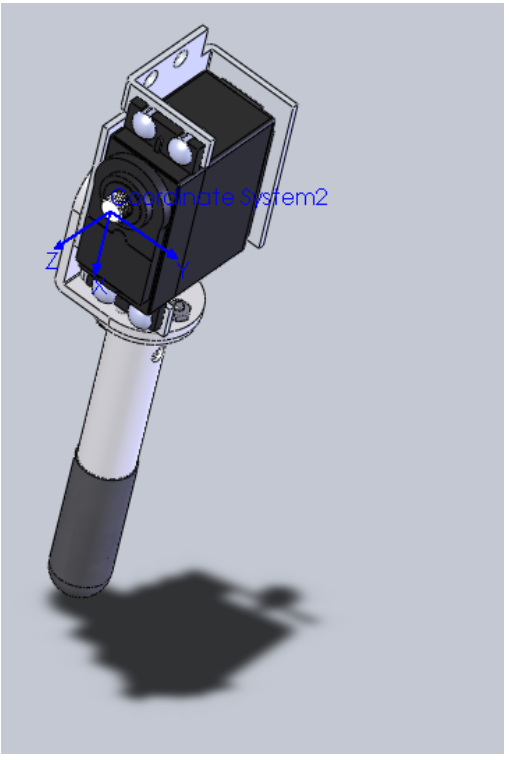

Figure B.6: Shows the coxa link for the two middle legs

$$
\begin{aligned}
IC &= \begin{bmatrix} 0.00001630 & 0.00000021 & -0.00000123 \\ 0.00000021 & 0.00009091 & 0.00000074 \\ -0.00000123 & 0.00000074 & 0.00008634 \end{bmatrix} \\
Pc &= \begin{bmatrix} 0.02214854 \\ -0.00199370 \\ -0.02222420 \end{bmatrix} \\
m &= 0.10218475
\end{aligned}
$$

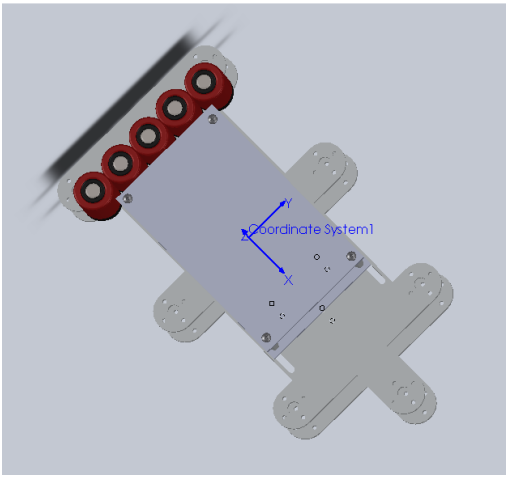

Figure B.7: Shows the assembled body of the hexapod robot

$$
\begin{aligned}
Ic &= \begin{bmatrix} 0.00150127 & 0.00001705 & -0.00042367 \\ 0.00001705 & 0.00368378 & -0.00002278 \\ -0.00042367 & -0.00002278 & 0.00357115 \end{bmatrix} \\
Pc &= \begin{bmatrix} -0.02218781 \\ 0.00015105 \\ 0.00410928 \end{bmatrix} \\
m &= 0.84121130
\end{aligned}
$$

Appendix C

Solutions to constraints

In the following, a small derivation of new parameters for the gait is presented. As described in section 18.4, the individual legs of the MR is subject to at least two constraints, when determining the AEP and PEP points. These two constraints were labelled length constraints and domain constraints. When the placement of the AEP and PEP points, given indirectly by the parameters v,θ and ϕ, breaks these constraints, it is necessary to change the gait. How the parameters of the gait should be changed depends directly on which one of the constraints that are broken. The first constraint that is considered is the length constraint.

C.1 Length constraint

As described in section 18.4 the length of the leg can be described by two different equations, one describing the AEP psition, and one describing the PEP position. These two equations are:

$$\begin{aligned} l_{AEP} &= \sqrt{\left(\cos\left(1/2\,\theta\right)x_n + \sin\left(1/2\,\theta\right)y_n + 1/2\,v\cos\left(-1/2\,\theta+\phi\right) - x_{coxa}\right)^2} + \cdots \\ &\cdots \sqrt{\left(-\sin\left(1/2\,\theta\right)x_n + \cos\left(1/2\,\theta\right)y_n + 1/2\,v\sin\left(-1/2\,\theta+\phi\right) - y_{coxa}\right)^2} \qquad \text{(C.1)} \\ l_{PEP} &= \sqrt{\left(\cos\left(1/2\,\theta\right)x_n - \sin\left(1/2\,\theta\right)y_n - 1/2\,v\cos\left(1/2\,\theta+\phi\right) - x_{coxa}\right)^2} + \cdots \\ &\cdots \sqrt{\left(\sin\left(1/2\,\theta\right)x_n + \cos\left(1/2\,\theta\right)y_n - 1/2\,v\sin\left(1/2\,\theta+\phi\right) - y_{coxa}\right)^2} \qquad \text{(C.2)} \end{aligned}$$

If a length constraint have been encountered, it is necessary to find a v which minimizes the length in such a way that the constraint remains unbroken. By replacing l_{AEP} and l_{PEP} in the

above equations, and solving for v, the following solutions are found: **AEP:**

$$
\begin{aligned}
v &= 2\sqrt{\left(\sin\left(1/2\,\theta\right)y_n+\cos\left(1/2\,\theta\right)x_n-x_{coxa}\right)^2\left(\cos\left(-1/2\,\theta+\phi\right)\right)^2}-\cdots\\
&\cdots\ \overline{2\,\sin\left(-1/2\,\theta+\phi\right)\left(\sin\left(1/2\,\theta\right)x_n-\cos\left(1/2\,\theta\right)y_n+y_{coxa}\right)}\cdot\cdots\\
&\cdots\ \overline{\left(\sin\left(1/2\,\theta\right)y_n+\cos\left(1/2\,\theta\right)x_n-x_{coxa}\right)\cos\left(-1/2\,\theta+\phi\right)}+\cdots\\
&\cdots\ \overline{\left(\sin\left(1/2\,\theta\right)x_n-\cos\left(1/2\,\theta\right)y_n+y_{coxa}\right)^2\left(\sin\left(-1/2\,\theta+\phi\right)\right)^2}+\cdots\\
&\cdots\ \overline{\left(2\,x_{coxa}x_n+2\,y_ny_{coxa}\right)\cos\left(1/2\,\theta\right)+\left(2\,x_{coxa}y_n-2\,y_{coxa}x_n\right)\sin\left(1/2\,\theta\right)}-\cdots\\
&\cdots\ \overline{{y_{coxa}}^2-{x_{coxa}}^2+aeplen^2-{x_n}^2-{y_n}^2}\\
&+\ 2\,x_{coxa}-2\,\sin\left(1/2\,\theta\right)y_n-2\,\cos\left(1/2\,\theta\right)x_n
\end{aligned}
\tag{C.3}
$$

$$
\begin{aligned}
v &= -2\sqrt{\left(\sin\left(1/2\,\theta\right)y_n+\cos\left(1/2\,\theta\right)x_n-x_{coxa}\right)^2\left(\cos\left(-1/2\,\theta+\phi\right)\right)^2}-\cdots\\
&\cdots\ \overline{2\,\sin\left(-1/2\,\theta+\phi\right)\left(\sin\left(1/2\,\theta\right)x_n-\cos\left(1/2\,\theta\right)y_n+y_{coxa}\right)}\cdot\cdots\\
&\cdots\ \overline{\left(\sin\left(1/2\,\theta\right)y_n+\cos\left(1/2\,\theta\right)x_n-x_{coxa}\right)\cos\left(-1/2\,\theta+\phi\right)}+\cdots\\
&\cdots\ \overline{\left(\sin\left(1/2\,\theta\right)x_n-\cos\left(1/2\,\theta\right)y_n+y_{coxa}\right)^2\left(\sin\left(-1/2\,\theta+\phi\right)\right)^2}+\cdots\\
&\cdots\ \overline{\left(2\,x_{coxa}x_n+2\,y_ny_{coxa}\right)\cos\left(1/2\,\theta\right)+\left(2\,x_{coxa}y_n-2\,y_{coxa}x_n\right)\sin\left(1/2\,\theta\right)}-\cdots\\
&\cdots\ \overline{{y_{coxa}}^2-{x_{coxa}}^2+aeplen^2-{x_n}^2-{y_n}^2}\\
&+\ \left(2\,x_{coxa}-2\,\sin\left(1/2\,\theta\right)y_n-2\,\cos\left(1/2\,\theta\right)x_n\right)\cos\left(-1/2\,\theta+\phi\right)\\
&+\ \left(2\,y_{coxa}-2\,\cos\left(1/2\,\theta\right)y_n+2\,\sin\left(1/2\,\theta\right)x_n\right)\sin\left(-1/2\,\theta+\phi\right)
\end{aligned}
\tag{C.4}
$$

PEP:

$$
\begin{aligned}
v &= -2\sqrt{\left(-x_{coxa}+\cos\left(1/2\,\theta\right)x_n-\sin\left(1/2\,\theta\right)y_n\right)^2\left(\cos\left(1/2\,\theta+\phi\right)\right)^2}+\cdots\\
&\cdots\ \overline{2\sin\left(1/2\,\theta+\phi\right)\left(-y_{coxa}+\cos\left(1/2\,\theta\right)y_n+\sin\left(1/2\,\theta\right)x_n\right)}\cdot\cdots\\
&\cdots\ \overline{\left(-x_{coxa}+\cos\left(1/2\,\theta\right)x_n-\sin\left(1/2\,\theta\right)y_n\right)\cos\left(1/2\,\theta+\phi\right)}+\cdots\\
&\cdots\ \overline{\left(-y_{coxa}+\cos\left(1/2\,\theta\right)y_n+\sin\left(1/2\,\theta\right)x_n\right)^2\left(\sin\left(1/2\,\theta+\phi\right)\right)^2}+\cdots\\
&\cdots\ \overline{\left(2\,x_{coxa}x_n+2\,y_ny_{coxa}\right)\cos\left(1/2\,\theta\right)+\left(2\,y_{coxa}x_n-2\,x_{coxa}y_n\right)\sin\left(1/2\,\theta\right)}-\cdots\\
&\cdots\ \overline{{x_{coxa}}^2-{y_{coxa}}^2+peplen^2-{y_n}^2-{x_n}^2}\\
&+\ \left(2\,\cos\left(1/2\,\theta\right)x_n-2\,x_{coxa}-2\,\sin\left(1/2\,\theta\right)y_n\right)\cos\left(1/2\,\theta+\phi\right)\\
&+\ \left(2\,\cos\left(1/2\,\theta\right)y_n-2\,y_{coxa}+2\,\sin\left(1/2\,\theta\right)x_n\right)\sin\left(1/2\,\theta+\phi\right) \qquad \text{(C.5)}\\
v &= 2\sqrt{\left(-x_{coxa}+\cos\left(1/2\,\theta\right)x_n-\sin\left(1/2\,\theta\right)y_n\right)^2\left(\cos\left(1/2\,\theta+\phi\right)\right)^2}+\cdots\\
&\cdots\ \overline{2\sin\left(1/2\,\theta+\phi\right)\left(-y_{coxa}+\cos\left(1/2\,\theta\right)y_n+\sin\left(1/2\,\theta\right)x_n\right)}\cdot\cdots\\
&\cdots\ \overline{\left(-x_{coxa}+\cos\left(1/2\,\theta\right)x_n-\sin\left(1/2\,\theta\right)y_n\right)\cos\left(1/2\,\theta+\phi\right)}+\cdots\\
&\cdots\ \overline{\left(-y_{coxa}+\cos\left(1/2\,\theta\right)y_n+\sin\left(1/2\,\theta\right)x_n\right)^2\left(\sin\left(1/2\,\theta+\phi\right)\right)^2}+\cdots\\
&\cdots\ \overline{\left(2\,x_{coxa}x_n+2\,y_ny_{coxa}\right)\cos\left(1/2\,\theta\right)+\left(2\,y_{coxa}x_n-2\,x_{coxa}y_n\right)\sin\left(1/2\,\theta\right)}-\cdots\\
&\cdots\ \overline{{x_{coxa}}^2-{y_{coxa}}^2+peplen^2-{y_n}^2-{x_n}^2}\\
&+\ \left(2\,\cos\left(1/2\,\theta\right)x_n-2\,x_{coxa}-2\,\sin\left(1/2\,\theta\right)y_n\right)\cos\left(1/2\,\theta+\phi\right)\\
&+\ \left(2\,\cos\left(1/2\,\theta\right)y_n-2\,y_{coxa}+2\,\sin\left(1/2\,\theta\right)x_n\right)\sin\left(1/2\,\theta+\phi\right) \qquad \text{(C.6)}
\end{aligned}
$$

These solutions are used to derive a new value for v. If no v values are able to satisfy both the AEP and PEP constraints, v is truncated to zero, and when looking at the domain constraints, only θ is modified. This entails that a new v can be found, if the length constraint is broken. The next constraint that needs to be taken into account is the domain constraint

C.2 Domain constraint

The other constraint which is taken into account is the domain constraint. As described in section 18.4, Each of the six legs are only allowed to have AEP and PEP points within a limited area, to prevent collisions with other legs. To avoid these collisions, each leg is assigned a certain domain limited by certain values of x. The x position of the PEP and AEP positions can be found by the ollowing equations:

$$x_{AEP}=\cos\left(1/2\,\theta\right)x_n+\sin\left(1/2\,\theta\right)y_n+1/2\,v\cos\left(-1/2\,\theta+\phi\right) \qquad \text{(C.7)}$$

$$x_{PEP}=\cos\left(1/2\,\theta\right)x_n-\sin\left(1/2\,\theta\right)y_n-1/2\,v\cos\left(1/2\,\theta+\phi\right) \qquad \text{(C.8)}$$

If the domain constraint is broken, even after v has been changed, when considering the length constraints, a new value of v should be calculated. This entails v is isolated in the

ablove equations.

$$v = -2\,\frac{-aepxpos + \cos\left(1/2\,\theta\right) x_n + \sin\left(1/2\,\theta\right) y_n}{\cos\left(-1/2\,\theta + \phi\right)} \tag{C.9}$$

$$v = 2\,\frac{-pepxpos + \cos\left(1/2\,\theta\right) x_n - \sin\left(1/2\,\theta\right) y_n}{\cos\left(1/2\,\theta + \phi\right)} \tag{C.10}$$

If these equations provide solutions still not able to comply with the imposed constraints, the only option left is to change the angle of the walk, θ. To do this, v in equation C.7 and C.8 is set to zero, and θ is isolated. This provides the following solutions:

AEP

$$\begin{aligned}\theta &= 2\,\arctan\left(\frac{x_{AEP}\,{y_n}^2 - x_n\sqrt{{x_n}^2{y_n}^2 - {y_n}^2{x_{AEP}}^2 + {y_n}^4}}{\left({x_n}^2 + {y_n}^2\right) y_n}\right.\\ &\left., \quad \frac{x_{AEP}\,x_n + \sqrt{{x_n}^2{y_n}^2 - {y_n}^2{x_{AEP}}^2 + {y_n}^4}}{{x_n}^2 + {y_n}^2}\right)\end{aligned} \tag{C.11}$$

$$\begin{aligned}\theta &= 2\,\arctan\left(\frac{x_{AEP}\,{y_n}^2 + x_n\sqrt{{x_n}^2{y_n}^2 - {y_n}^2{x_{AEP}}^2 + {y_n}^4}}{\left({x_n}^2 + {y_n}^2\right) y_n}\right.\\ &\left., \quad \frac{x_{AEP}\,x_n - \sqrt{{x_n}^2{y_n}^2 - {y_n}^2{x_{AEP}}^2 + {y_n}^4}}{{x_n}^2 + {y_n}^2}\right)\end{aligned} \tag{C.12}$$

PEP

$$\begin{aligned}\theta &= 2\,\arctan\left(\frac{-x_{PEP}\,{y_n}^2 + x_n\sqrt{{x_n}^2{y_n}^2 - {y_n}^2{x_{PEP}}^2 + {y_n}^4}}{\left({x_n}^2 + {y_n}^2\right) y_n}\right.\\ &\left., \quad \frac{x_{PEP}\,x_n + \sqrt{{x_n}^2{y_n}^2 - {y_n}^2{x_{PEP}}^2 + {y_n}^4}}{{x_n}^2 + {y_n}^2}\right)\end{aligned} \tag{C.13}$$

$$\begin{aligned}\theta &= 2\,\arctan\left(\frac{-x_{PEP}\,{y_n}^2 - x_n\sqrt{{x_n}^2{y_n}^2 - {y_n}^2{x_{PEP}}^2 + {y_n}^4}}{\left({x_n}^2 + {y_n}^2\right) y_n}\right.\\ &\left., \quad \frac{x_{PEP}\,x_n - \sqrt{{x_n}^2{y_n}^2 - {y_n}^2{x_{PEP}}^2 + {y_n}^4}}{{x_n}^2 + {y_n}^2}\right)\end{aligned} \tag{C.14}$$

This entails that solutions are obtained in both the case where the length constraint and the domain constraint are specified. It is however also seen that there exists two solution for both the length constraint, when solving for v, and the domain constraint, when solving for θ. These spurious solutions are however easy to detect, as the length constraint always gives very big solutions, and the angle constraint always provide solutions in a different region of the unit circle, than the leg is able to access. This again entails that a solution to new v or θ values can be found, when a constraint is broken.

Appendix D

Camera calibration

In the following, a more thorough description of the camera calibration mentioned in section 15 is conducted. As described, the reference to the MR should be given in the real world, it is necessary to make a description of how the world representation of $\surd$ can be computed when a coordinate in the image is given as the current destination. This can however be quite problematic as a degree of freedom is lost when moving from the 3d world to the image plane. To be able to reconstruct this missing dimension, one needs use additional information regarding the position of points which are observed. This information is found by using the assumption that all points lie on a ground plane. To be able to use this information, a so called calibration of the camera is needed.

D.1 Camera calibration

The procedure described in the following chapter is mainly based on the procedure described in section [2, p82-88], but also from [1, p129-131]. This calibration is composed of finding the matrix, which is able to map real world coordinates **C**=[x,y,z,1]^T to image coordinates c=(u,v). According to [2, p86-88] this mapping can be described by means of a matrix $\tilde{\mathbf{P}}$, called the projection matrix.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \tilde{\mathbf{P}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{D.1}$$

The image coordinates u and v can be found as u=x1/x3, and v=x2/x3.
By insertions, and writing $\tilde{\mathbf{P}}$ as composed of three row vectors, the following is obtained:

$$\begin{bmatrix} u \cdot x_3 \\ v \cdot x_3 \\ x_3 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{P}}_1 \\ \tilde{\mathbf{P}}_2 \\ \tilde{\mathbf{P}}_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{D.2}$$

According to [2] one of the simplest ways of deriving this calibration matrix is by making a least square optimzation. If a collection of points P_i is given in world coordinates, along with their image projections (u_i, v_i) one can define the image error as being the squared difference between the known image coordinates, and the projection of the real world ones. If equation D.2 is used, this error can be described in the following way:

$$\epsilon = \sum \left((\tilde{\mathbf{P}}_1 \cdot \mathbf{C} - (\tilde{\mathbf{P}}_3 \cdot \mathbf{C}) \cdot u_i)^2 + (\tilde{\mathbf{P}}_2 \cdot \mathbf{C} - (\tilde{\mathbf{P}}_3 \cdot \mathbf{C}) \cdot v_i)^2 \right) \tag{D.3}$$

It can be shown that this least square solution has an analytical solution. If A is a matrix of the following form:

$$A = \begin{bmatrix} -X_i & -Y_i & -Z_i & -1 & 0 & 0 & 0 & 0 & u_i X_i & u_i Y_i & u_i Z_i \\ 0 & 0 & 0 & 0 & -X_i & -Y_i & -Z_i & -1 & v_i X_i & v_i Y_i & v_i Z_i \\ \dots & & & & & & & & & & \end{bmatrix} \tag{D.4}$$

and B is a matrix of the following form:

$$B = \begin{bmatrix} -u_i \\ -v_i \end{bmatrix} \tag{D.5}$$

And finally if one partitions the $\tilde{\mathbf{P}}$ matrix into the following vector:

$$X^T = \begin{bmatrix} \tilde{\mathbf{P}}_{1,1} & \tilde{\mathbf{P}}_{1,2} & \tilde{\mathbf{P}}_{1,3} & \tilde{\mathbf{P}}_{1,4} & \tilde{\mathbf{P}}_{2,1} & \tilde{\mathbf{P}}_{2,2} & \tilde{\mathbf{P}}_{2,3} & \tilde{\mathbf{P}}_{2,4} & \tilde{\mathbf{P}}_{3,1} & \tilde{\mathbf{P}}_{3,2} & \tilde{\mathbf{P}}_{3,3} \end{bmatrix} \tag{D.6}$$

The solution to the above, can be found by means of the following equation:

$$X = \left(A^T \cdot A\right)^{-1} \cdot A \cdot B \tag{D.7}$$

This entails, that given a collection of points in world coordinates, and their image representations, one can find the projection matrix $\tilde{\mathbf{P}}$.

D.1.1 Finding the projection matrix

To find the projection matrix, one needs a set of real world coordinates and, a set of real image coordinates. One can use a so called calibration target, as recommended by [2, p86]. As no calibration target were present a simple dot map were drawn on a big piece of paper, and a picture were taken with the MR. The dot distribution, can be seen from picture D.1, and the image can be seen from picture D.2. Plase note, that the "crookedness" of Figure D.2

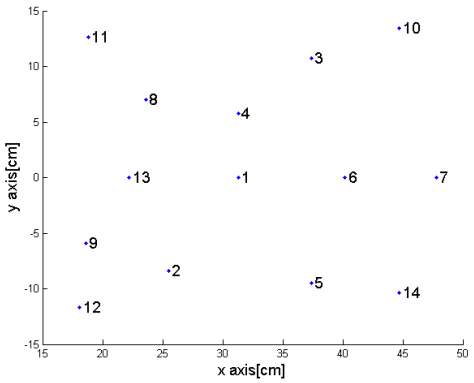

Figure D.1: Shows the points on the dot map

Figure D.2: Shows the image projection of the points on the dot map

is a result of inaccuracy of the camera mounting, and not the cause of an unprecise position of the robot. These points can also be seen from table D.1 From these figures and tables,

Point	Image coordinate[-]	Real world coordinate[cm]
1	(179 102)	(31.3 0)
2	(286 137)	(25.5 -8.4)
3	(77 72)	(37.4 10.7)
4	(114 102)	(31.3 5.8)
5	(273 67)	(37.4 -9.5)
6	(180 63)	(40.2 0)
7	(181 35)	(47.8 0)
8	(79 149)	(23.6 7)
9	(270 197)	(18.6 -5.9)
10	(70 52)	(44.7 13.4)
11	(0 189)	(18.8 12.6)
12	(18.1,-11,7)	(-,-)
13	(175 159)	(22.2 0)
14	(267 44)	(44.7 -10.4)

Table D.1: Shows the points on along with the image coordinates of the dot map.

it can be seen that one points have not been mapped from the dot map to the image, point

12. But as only 6 points are needed[1] to make the projection, the remaining 13 points are more than sufficient. By using these points, with the the calibration matrices, the calibration matrix have been found to be:

$$\tilde{\mathbf{P}} = \begin{bmatrix} 6.8336 & -17.7735 & 3.0099 & 138.6582 \\ -5.7147 & 1.2876 & 2.7192 & 395.7192 \\ 0.0347 & 0.0080 & 0.0226 & 1.0000 \end{bmatrix} \tag{D.8}$$

Appendix E

Test of inverse kinematics

In section 6 an inverse kinematic model were derived. This model should be able to determine the required angles of the coxa, femur and tibia joints, if an end point position is provided. In the following, a small test of this model is made. The test is conducted in the following way: An end point coordinate is passed to the inverse kinematics equations, which passes the corresponding coxa, femur and tibia angles. These angles are then passed to the servo motors of the left middle leg of the robot, and the obtained end point position is then measured.
The test were conducted in three steps. One measureing the deviations in the y direction, one measureing the deviations in the x direction, and finally one in the z direction. The obtained results can be seen from table E.1-E.3.

E.1 First test - x axis

The first that is conducted is on the x axis, here the y axis is held constantly at 10, and the z axis is held at -16. The results from this experiment can be seen from table E.1

Test	Expected x position [cm]	Real world coordinate[cm]	deviation [%]
1	-8	-8.5	12.5
2	-6	-6.5	8.3
3	-4	-4	-2.5
4	-2	-2	0
5	0	0	0
6	2	2.1	5
7	4	4.3	7.5
8	6	6.7	11.7
9	8	9	12.5

Table E.1: Shows the measurements of the displacement of the inverse kinematics model

E.2 Second test - y axis

The second conducted test were in the y direction. Here the x axis were held at 0, and the z axis at -16. The results from this experiment can be seen from table E.2

Test	Expected y position [cm]	Real world coordinate[cm]	deviation [%]
1	1	0.8(Minimum)	20
2	2	1.8	10
3	3	2.9	3.3
4	4	4.0	0
5	5	5.0	0
6	6	6.0	3.3
7	7	7.0	0
8	8	8.1	1.3
9	9	9.2	2.2
10	10	10.3	3.3
11	11	11.3	2.7
12	12	12.4	3.3
13	13	13.6	4.6
14	14	14.7	5.0
15	15	15.5	3.3
16	16	17(Maximum)	6.3

Table E.2: Shows the measurements of the displacement of the inverse kinematics model

E.3 Third test - z axis

The final test is in the z direction. The x axis and y axis are held at respectively 0 and 10 cm, while the height of the end effector is varied. The results from this experiment can be seen from table E.3

Test	Expected xposition [cm]	Real world coordinate[cm]	deviation [%]
1	-16	-16.3	1.9
2	-15	-15.4	2.7
3	-14	-13.8	1.4
4	-13	-12.6	3.1
5	-12	-11.8	1.7
6	-11	-10.7	4.5

Table E.3: Shows the measurements of the displacement of the inverse kinematics model

E.4 Conclusion

From these tests, it can be seen that generally the correspondence between the inverse kinematic model, and the real world coordinates are quite good. It is also seen, that the differences are largest, when considering values close to the maximum allowed. The largest deviations are seen when considering the x axis positions, and it is also seen, that the errors grow, as the end effector is moved from the centre. This might suggest, that the coxa servo needs to be re-calibrated, but this possibility is not investigated further. It is however clear, that the robotic platform suffers quite severly from mechanical uncertanties.

Appendix F

Servo tests

In the following section, two tests of the servo motors are conducted. The first one concerns the hypothesis that the load on the servo decreases the angular velocity of the servo. The second test concerns the power consumption of the servo.

F.1 Test 1 - Load test of servo

In the following, a test of how a load affects the speed of a servo motor is made. The hypothesis is that a load on the servo decreases the velocity, and thus the time before the servo reaches it's goal position increases. To test this hypothesis, a small test setup is made in the laboratory. This test can be seen from figure F.1 A wheel, with 8 nails, placed on the

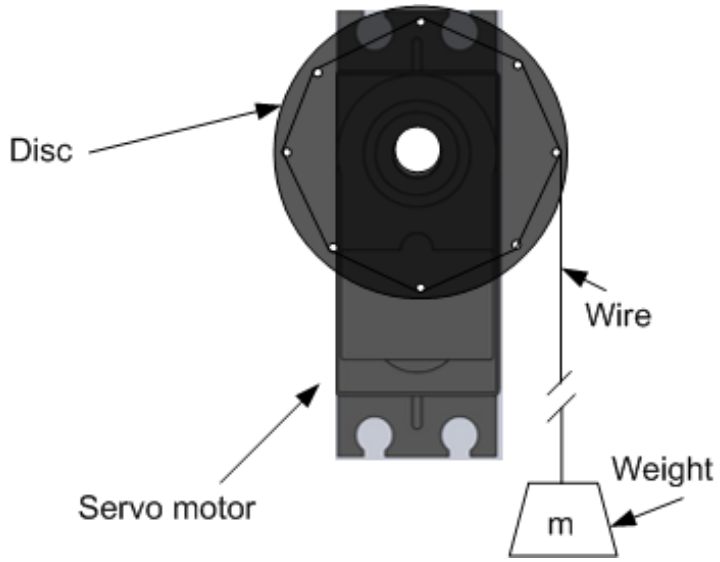

Figure F.1: Shows the test setup of the servo motor load test

peripheral of a circle, with a radius of 1.8 cm is placed on the servo. A string of fishing line were attached to one of the nails in one end, and to a series of weights on the other.
The servo motor is controlled by means of the position of an internal potentiometer. The servo motor were opened, and a probe were attached to the feedback potentiometer situated inside of the case. Now the servo were set to rotate from one extreme position to the other, while recording the potentiometer position. Six different of loads were attached to the end of the fishing line. In table F.1 the weights and the corresponding load torques can be seen. The recorded positions can be seen from figure F.2 From this figure it is seen, that the time

Test	Weight[g]	Torque
0	0	0
1	200	0.0354
2	400	0.0707
3	600	0.1061
4	800	0.1414
5	1100	0.1944
6	1300	0.2298

Table F.1: Shows the six errors weights which were used to load the servo, and the corresponding torques.

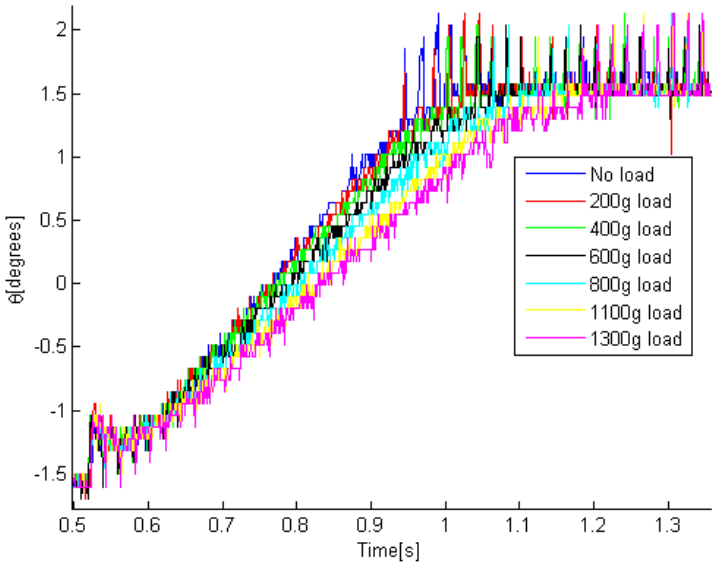

Figure F.2: Shows the obtained results of the servo motor measurements

for the servo to reach the desired angle in fact is increased with the load. It is however also seen that there is a lot of noise on the measurements, with big spikes. To find the decrease in motor speed, the obtained signals were smoothed, and discretely differentiated. These velocities can be seen from figure F.3. Another interesting plot can be seen from F.4, where

the maximum velocities are plotted against the load torques. From this figure it is seen that

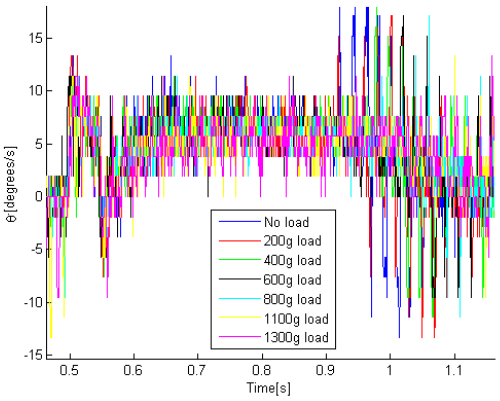

Figure F.3: Shows the velocity of the servo motor, when load torques are applied

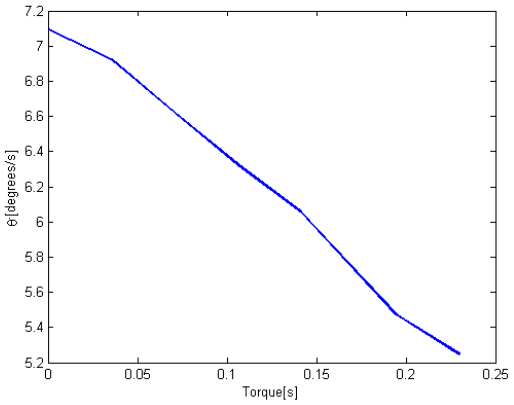

Figure F.4: Shows the velocity of the servo motor as a function of the load torques

there exists an almost linear relationship between the torques and maximum velocities. This linear relationship can be approximated with a line, with the following equation:

$$\omega(k)_{max} = 7.1574 - 8.2921\tau_l \tag{F.1}$$

The results of this section is used to make the dynamic model of the servo motors, described in section 7.1.

F.2 Test 2 - power consumption of servo

The second test is about the power consumption of the servo motor. As described in section 7, a way of verifying the model is needed. In this section, the legs of the MR were treated as manipulators, and by using the iterative newton euler approach, it is possible to find the torques needed to move the servo motors. As the current delivered to the servo motors should be proportional to the amount of power dispatched in the motor, and as the generated torque should be proportional to the consumed power, a proportional relation between torque and current is proposed. This entails that a current probe is attached to the wire providing the power to a coxa servo, which is set to rotate from -pi/4 to pi/4 radians. The same is done in the simulation, and the absolute value of the torque is obtained. When these two plots are superimposed, the figure from F.5 is obtained:

From this figure it is seen, that in the beginning, a somewhat identical behaviour can be seen, which describes the same tendencies. However when the servo starts to decellerate, this match completely stops. The curve of the load torque from the dynamic model has a slow decline, while the current to the servo motor abruptly stops. It should be noticed that

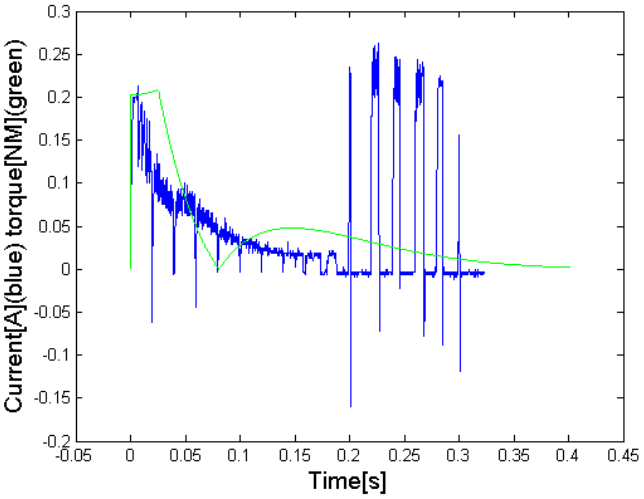

Figure F.5: Shows a plot of the torque of the dynamic model, and the current delivered to the servo motor

when the smooth behaviour of th esevo motor stops, the servo is still moving. It seems that it is being stopped by the big spikes seen on the right side of the graph. This entails that no conclusion regarding whether or not the dynamic model is correct can be reached.

Appendix G

Verification of simulation model

In the following chapter, the verification of the MR model is conducted. As described in section 10 a formal verification cannot be carried out, as no sensors are attached to the MR. This entails, that the model cannot be verified as being a model of the hexapod robot used in this project, but only as beloning to a class of mechanical systems, which behaves in a fashion which could be identified to be a Hexapod robot. To make this verification, a series of simulations have been carried out. Each of these tests will come with a description of the steps involved, a description of the expected behaviour, and finally a conclusion As described in section 7, the dynamic model is generally composed of two parts. A free body and six almost identical manipulators. As the manipulators have somewhat complex dynamics, the verification process is started by simulations of a single manipulator.

G.1 Dynamic verification of Manipulator

To determine if the dynamics of the manipulator are correct, two tests are carried out. One to show the important forces/velocities acting, when a motion in the x-y plane is conducted, and one to show the same factors, when a motion in the X-Z plane is conducted.

G.1.1 Step applied to coxa joint angle

Description

In the first simulation, the end effector speed, position base forces, and joint torqes are tested. A step from 0 to $\pi/2$ is applied to the coxa joint at time t=0.

Expected behaviour

In this simulation, it is expected that the end point position will trace the peripheral of a circle in the X-Y plane. The speed should be large in the negative x direction, and thereafter negative in the y direction. A z axis velocity should not be seen, unless the Tibia joint is forced to "open" due to strong centrifugal forces. When regarding the torques, a torque acting on the Coxa joint should be seen, a small torque torque should act on the Femur joint, due to gravity, and a small force should be seen on the Tibia joint, again due to gravity. One could expect, that a small torque is also induced from the motion, as the Tibia joint is forced to "open" at high velocities. The force acting on the base is quite hard to predict, as it is composed of centrifugal, coriolis and gravital components. It is suspected that the gravitational force is acting in the negative z axis(As normal), and that a force is acting in the positive x, and then negative y direction, as this would be the in the direction of the end effector. The magnitude of this force should be dependent on the velocity, with parts dependent on the acceleration of the legs.

Simulation

The obtained simulation results can be seen from figure G.1 to G.4

Conclusion

From the obtained results, it is seen that the endpoint position traces the arc of a circle. It is also seen that only velocities in the X and Y directions can be seen, which again was as predicted. From the plots of the torque, it is seen that the biggest load is seen on the coxa servo, and small loads are seen on the Femur and Tibia servos. From the force plots it is seen that the obtained force is pointing as expected. This entails that the expected behaviour of the system was also seen in the simulation

G.1.2 Step applied to Femur joint angle

Description of test

In the second simulation, a step from 0 to $\pi/3$ is applied to the Femur joint at time t=0.

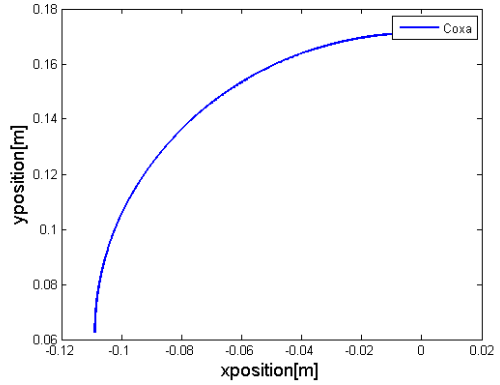

Figure G.1: Shows the position of the end effector

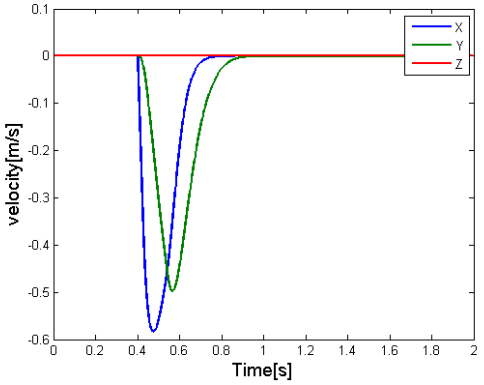

Figure G.2: Shows the velocity of the end effector

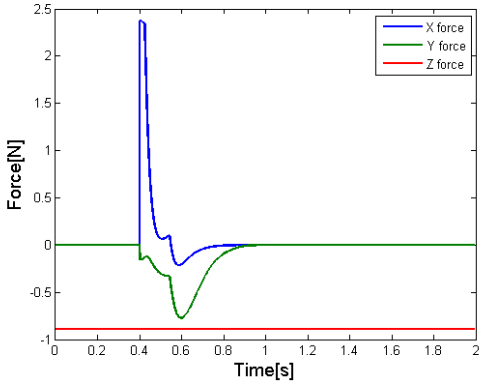

Figure G.3: Shows the base forces acting on the MR

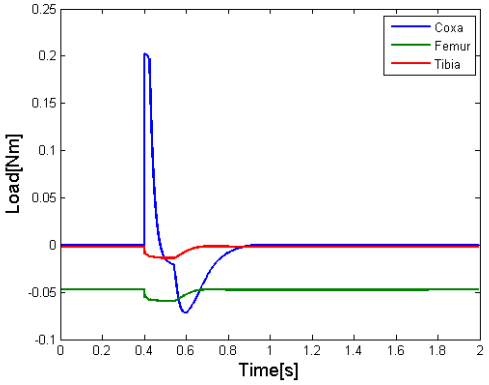

Figure G.4: Shows the torques acting on the joints

Expected behaviour

In this simulation, it is expected that the end point position will trace the peripheral of a circle in the Y-Z plane. The speed should be large in the positive z direction, and thereafter positive in the y direction. An x axis velocity should not be seen. When regarding the torques, a torque acting on the Femur joint should be seen, no torque ahould act on the Coxa joint,, as it is located perpendicular to both gravity and the forces acting on the Femur and Tibia joints, but a force should be seen on the Tibia joint, again due to centrifugal forces. And finally, a force acting on the base, in the direction of the end effector should be seen, which, as in the first simulation is dependent on the velocity and acceleration of the joints.

Simulation

The obtained simulation results can be seen from figure G.5 to G.7

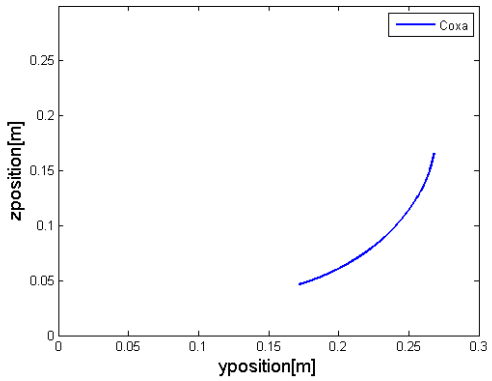

Figure G.5: Shows the position of the end effector

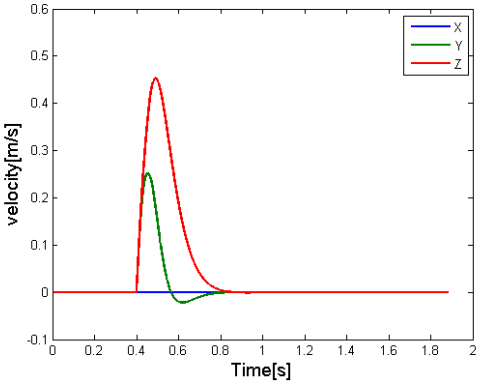

Figure G.6: Shows the velocity of the end effector

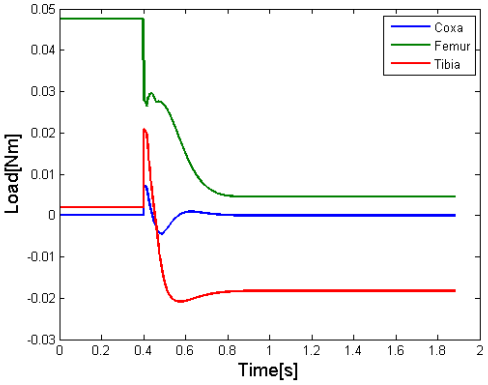

Figure G.7: Shows the torques acting on the joints

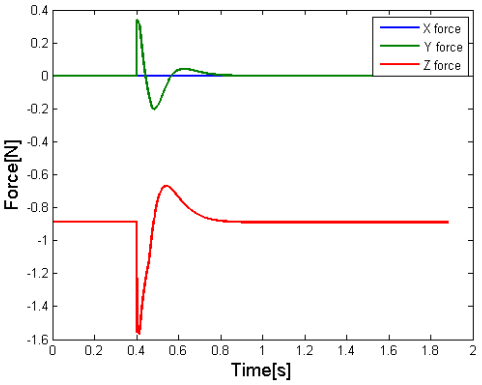

Figure G.8: Shows the base forces acting on the MR

Conclusion

From the obtained figures it is seen that the position traces the arc of a circle in the Y-Z plane, with velocities as described in the expected bahaviour. It is seen that a torque acting on the Femur joint is present, and that almost no load torque is affecting the coxa servo. It is also seen that the force acting on the base dependent on the acceleration and velocity can be seen.

G.2 Dynamic verification of complete model

Now that the manipulators have been verified, it is time to determine if the collected model, composed of the free body, and the 6 manipulators behave as expected, and the ground reaction force. In the last three simulations, the dynamics of the manipulators were verified. Now that a total of 6 manipulators, combined with a free body should be tested, the amount of graphs would be huge, and it would be difficult to conclude if the MR acts as suspected, if the same form of documentation were used. This entails that the form of the documentation

is changed to be composed of plots generated from the kinematic model, combined with the position and orientation of the centre of the robot, and plots showing other important aspects of the tests.

G.2.1 Drop MR on the ground

Description of test

The MR is dropped from a distance of 25 cm above the ground, and it is seen how it is stabilized on the ground

Expected behaviour

When no tangential forces act on the MR, it is suspected that the MR will strike the ground, with all legs at the same time. As the centre of gravity is not completely aligned with the centre of the MR coordinate system, it is suspected that a very small angular acceleration will occur, which quickly will be absorbed by the ground. This comes from the fact, that the hind legs support the battery, and thus will carry more weight.

Simulation

The obtained simulation results can be seen from figure G.9 to G.12

Conclusion

From these figures, it is seen that the MR falls to the ground with a linearly increasing velocity, until it hits the ground. When it hits, it is almost immediately stabilized by the ground, and no oscillations occur. The rotations are close to zero and all ground contact forces occur at almost the same time.

G.2.2 Drop with an angle

Description of test

This test is quite similar to the previous one. The MR is dropped from a distance of 25 cm above ground, but this time, with a start rotation about the x axis of pi/6 radians.

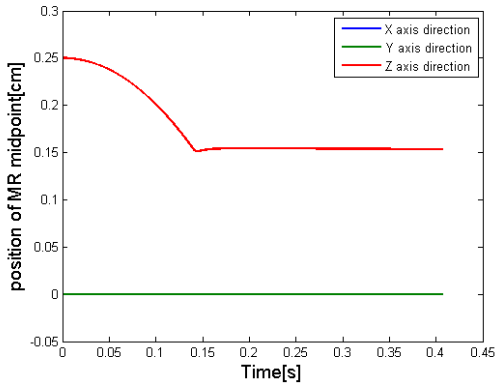

Figure G.9: Shows the position of the end effector

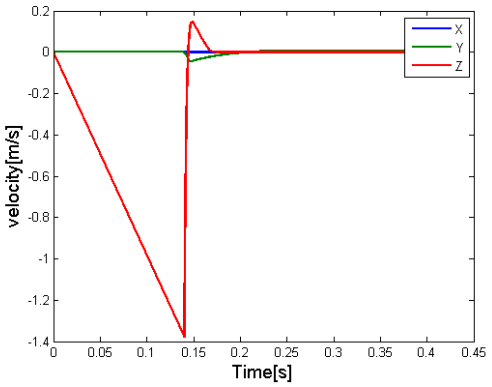

Figure G.10: Shows the velocity of the end effector

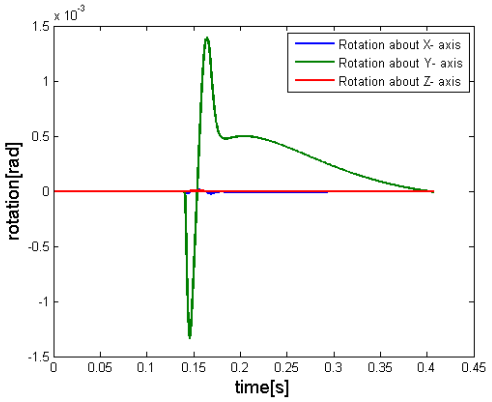

Figure G.11: Shows the base forces acting on the MR

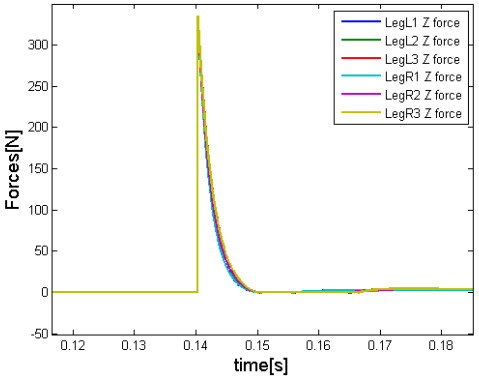

Figure G.12: Shows the torques acting on the joints

Expected behaviour

It is suspected that the MR will strike the ground, with the left middle leg, then a rotation will occur, and the MR will strike the ground with the middle right leg, althoug another rotation, again due to the difference between the position of the centre of mass and the centre of the robot will also occur.

Simulation

The obtained simulation results can be seen from figure G.13 to G.16 To ease the interpretation of these graphs, a set of poses generated from the kinematic model is provided. These can be seen from figure G.17-G.19

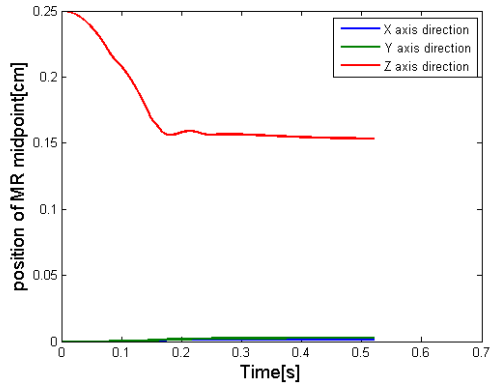

Figure G.13: Shows the position of the end effector

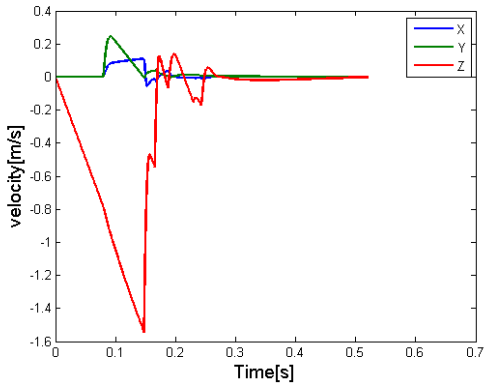

Figure G.14: Shows the velocity of the end effector

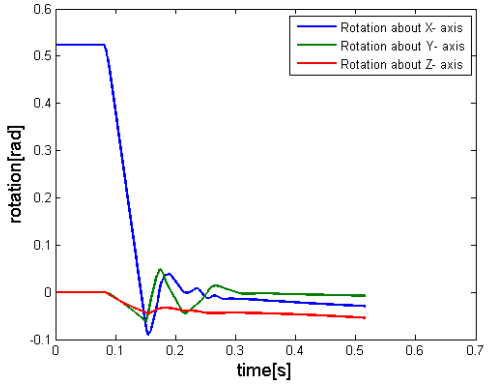

Figure G.15: Shows the base forces acting on the MR

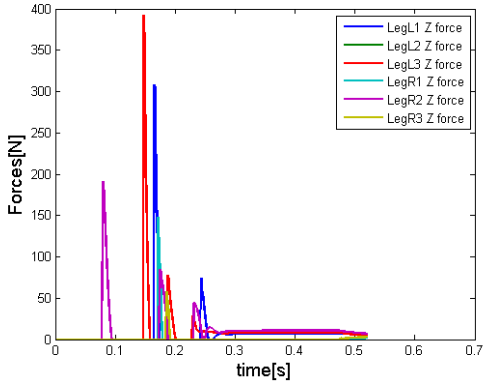

Figure G.16: Shows the torques acting on the joints

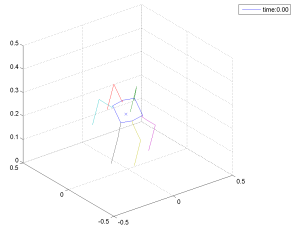

Figure G.17: t=0s

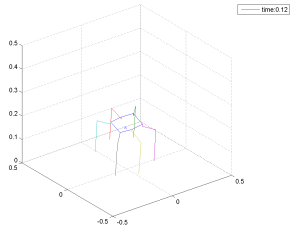

Figure G.18: t=0.12s

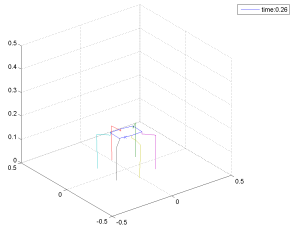

Figure G.19: t=0.26s

Conclusion

From these figures it is seen that the the MR strikes starts with a rotation of pi/6 around the x axis. It is then subject to a constant acceleration, which pulls it down. Then the left leg strikes the ground, and a rotation starts to occur. Then the other legs of the MR hits the ground, and a small oscillation occurs before the MR finally settles. It is actually seen from the force diagram, that the end effector force acting on the left middle leg becomes zero just after it hits. This can only be explained by the fact that the leg bounces on the surface. This could be a symptom of that the simulation has not been conducted with a sufficiently small

time step. This will however be discussed in section G.3.

G.2.3 One leg lifting

Description of test

In this test, it is shown what happens when one of the legs are extended to support the full weight of the MR. at time t=0.2, the middle left leg is extended, so that it touches the ground, and carries the weight of the MR.

Expected behaviour

It is expected that the leg, which strikes the ground, at first only is subject to a large amount of pressure, and while the force on the other two legs on the left side decreases. When the leg is supporting the full weight of the left side, a rotation about the z axis will occur, and the left side will begin to rise.

Simulation

The simulations conducted, can be seen from figure G.20-G.23. As in the previous section, some poses of the has been constructed using the kinematic model. These poses can be seen from figure G.24 to G.26.

Conclusion

It is seen from the plots of the MR poses, that the model does in fact picture the MR lifting one side of the body with its leg. It is seen that the MR is stable on three legs. The plots of the position and rotations are also as expected, but the forces acting on the end effector of the middle left leg are however not as expected. It is seen that many rapid changes occur. It is believed that these oscillations are caused by the feedback mechanism between the dynamic leg model, and the servo motor model. When a high load is applied to the servo motor, the maximum attainable velocity is limited, but as the load is directly dependent on this velocity, such oscillatory behaviour can be seen. Besides the oscillating forces and torques, the general behaviour of the model is as expected.

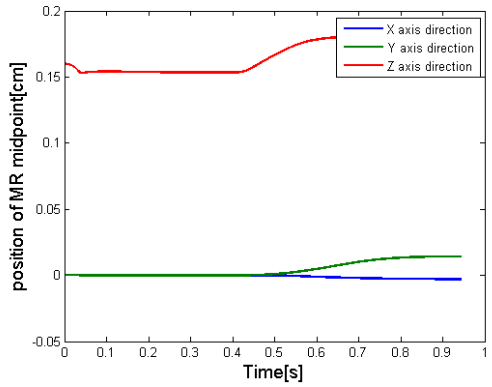

Figure G.20: Shows the position of the centre of the robot

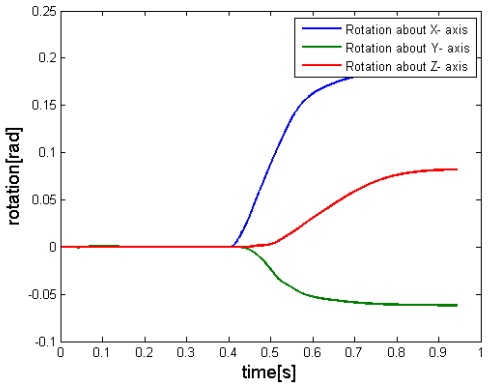

Figure G.21: Shows the rotation around the centre of the robot

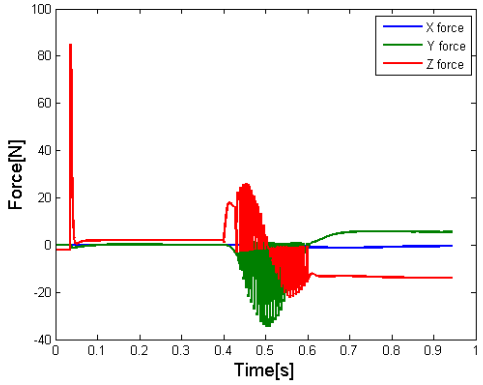

Figure G.22: Shows the force acting on the end effector

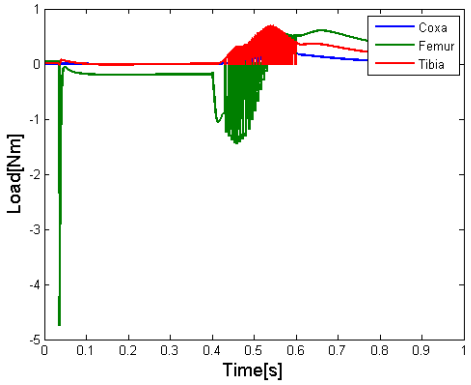

Figure G.23: Shows the load torque on the middle left leg

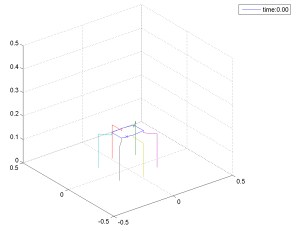

Figure G.24: t=0s

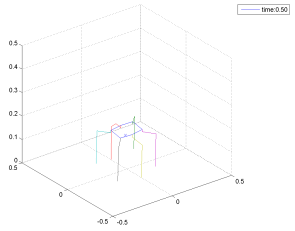

Figure G.25: t=0.5s

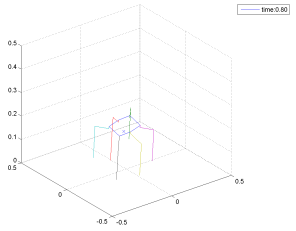

Figure G.26: t=0.8s

G.2.4 Walk

Description of test

To make a simulation, which shows that the model can be used for simulating the gait of a hexapod robot. A simple gait is simulated. The gait simulates a frontal walk, which should be able to travers 10 cm each gait cycle, with no movement in the y direction. The first 0.2 seconds of the gait is an initialization, where the legs of the MR are put in the correct start

positions. Then a walk sequence is initiated, with a switch, where First leg L1 L3 and R2 are in the air, while L2 R1 R3 moves the weight of the MR forward. Then another shift phase, and then L1 L3 and R2 moves the weight of the MR forward, while L2 R1 R3 returns.

Expected behaviour

The first 0.2 seconds of the simulation should not be taken into account, as it is an initialization phase. Then a switch phase is initiated, where no motion should occur. After this period, it is expected that the MR moves in the x direction, with a speed of 10cm/step, with no movement in the y direction.

Simulation

The simulation of the gait extends for more than 4 seconds. To show the movement of the MR during these four gait phases, 4 kinematic configurations are shown from figure G.27 to G.30. On figure G.27 the startposition is shown. On figure G.28 a shot of the first phase is found, and on figure G.29 a shot of the second phase is shown. On figure G.30 the end position is shown

Besides these plots of the kinemtaics, describing the gait, three plots are provided. A plot of the position of the centre of the MR is plotted on figure G.31 a plot is made of the gait phases on Figure G.32. And finally, a plot of the legforces involved can be seen from figure G.33 zommed in on a section on figure G.34. The final plot that is shown is a plot of the rotation of the MR.

Conclusion

From the plots it is seen that several of the expectations have been fulfilled. The centre of the robot moves in the x direction, while almost no movement is seen in the y and z direction. One thing that however is not as expected is the results obtained when consdering the rotation around the centre of the MR. From figure G.35 it is seen that minor oscillations occur. The cause of these oscillations will be considered in the limitations of the model.

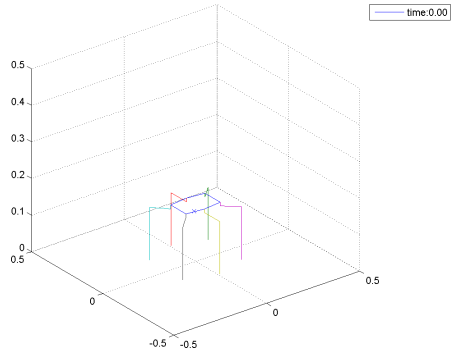

Figure G.27: Shows a snapshot of a simulation of a simple gait, This shot shows the beginning of the gait

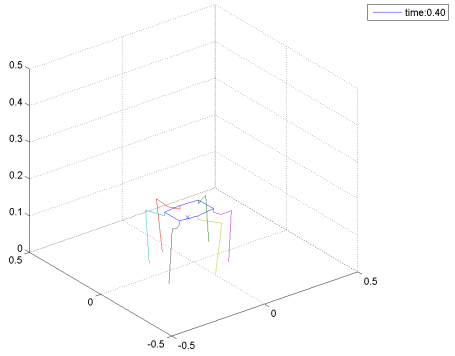

Figure G.28: Shows a snapshot of a simulation of a simple gait, this shot shows phase 1 of the gait

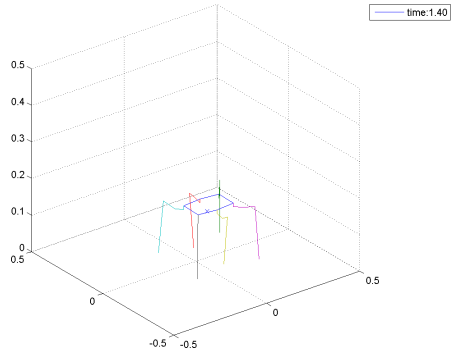

Figure G.29: Shows a snapshot of a simulation of a simple gait, this shot shows phase 2 of the gait

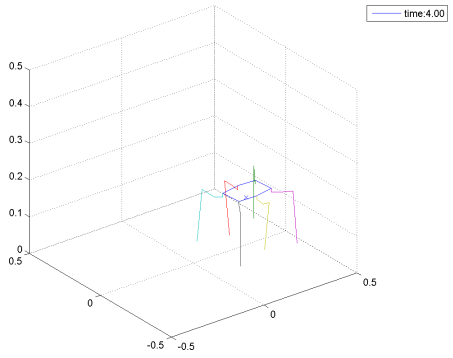

Figure G.30: Shows a snapshot of a simulation of a simple gait, this shot shows the end position of the MR

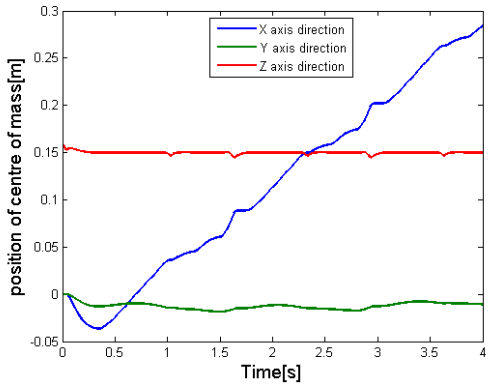

Figure G.31: Shows the position of the centre of the robot, as a function of time

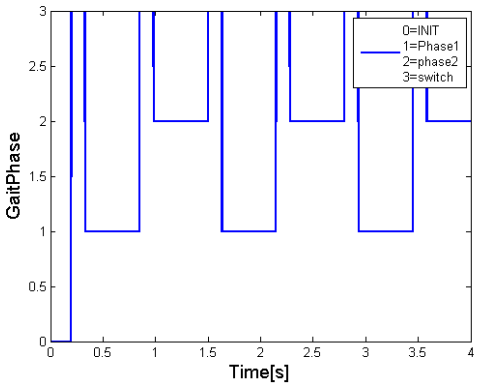

Figure G.32: Shows the gait phases of the simulated gait

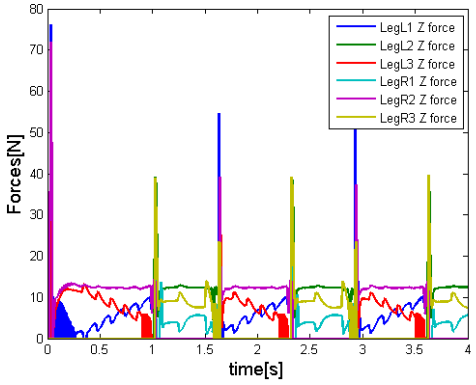

Figure G.33: Shows the end point forces acting on the MR over the entire timeframe

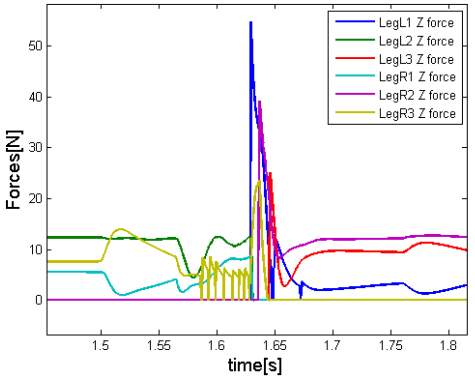

Figure G.34: Shows a section of figure G.33

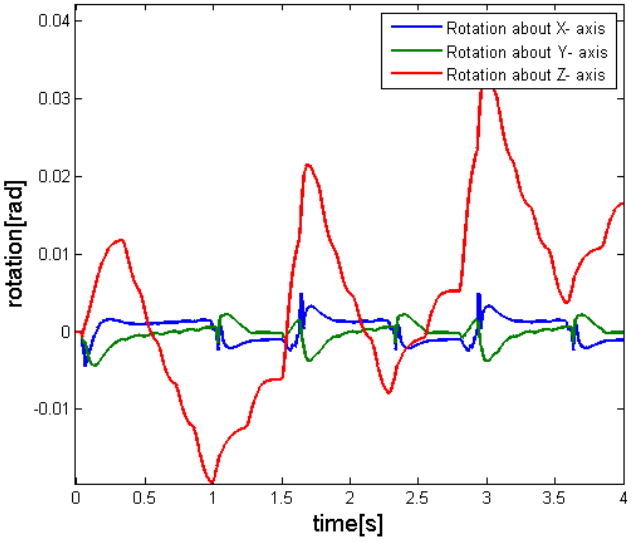

Figure G.35: Shows the rotation of the centre of the robot

G.3 Known limitations of the model

While conducting the simulations, two problems with the model structure were found. Both concerning the ground reaction force. In section 7.3 , concerning the ground reaction force two choises were made. The first choise that were made was model the force acting on the MR from the ground, as a spring damper system. This had the obvious benefit of being a simple way of modelling ground contacts. This choise has on the other hand resulted in that the model needs to run with a sampletime Ts=0.0005 if an unstable system should not be obtained. This again entails that it takes almost 1 minute in real time to simulate one second of model time. Even at this sample time, the model can become unstable if subject to high velocities or rotations.
The second choise that was made, was to simulate the frictional force as a viscous friction only. This has entailed, that when a walk simulation was introduced, the orientation of the MR had minor oscillations. These oscillations occur due to the way friction is implemented in the model. As described in section 7.3, the friction with the surface is only modelled as a viscous friction. As one of the sides of the MR always has two endeffectors on the ground, and the other only one, the amount of friction will be unbalanced, and a small rotation will occur. This unbalance is however evened out, when the side with two legs on the ground changes. To remove this effect, another type of friction should be implemented. These unbalances are however quite small, and gives only rise to angle deviations around the magnitude of $\pm 1^{\circ}$, and as another friction models, would require more computational power, it is assesed that the benefits are outweighed by the drawbacks. This entails that the friction model is kept as a strictly viscous friction.

G.4 Conclusion

From the conducted tests it can be seen that the outputs from the dynamic model generally matches the expected behaviour of the hexapod robot. As it is impossible to directly verify the model based on nothing but simulations, and "common sense" the reached conclusion is not that the model is able to behaviour of the mobile robot. Instead it is concluded that the model is able to describe the behavour of a sixlegged walking machine. If the model were to be made over again, a new friction models could be utilised to obtain better results regarding rotation.