THESIS FOR THE DEGREE OF MASTER OF SCIENCE IN
COMPLEX ADAPTIVE SYSTEMS

MODEL-BASED CONTROL OF A HUMANOID BIPEDAL ROBOT

JOHAN KRISTENSON

Department of Applied Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2008

Model-Based Control of a Humanoid Bipedal Robot

JOHAN KRISTENSON

Department of Applied Physics
Chalmers University of Technology
412 96 Göteborg, Sweden
Telephone: +46 (0)31 772 1000

Chalmers reproservice
Göteborg, Sweden, 2008

Abstract

This master thesis is concerned with inverse kinematics for humanoid bipedal robots in general and the robot ZORC in particular. The possibility of using simulation software, specifically CADGene and Sigel, when searching for bipedal gaits is examined. Furthermore, pressure sensors are tested for finding the centre of mass of a bipedal robot. The use of pressure sensors for robot balancing is also investigated. It is concluded that Sigel is not suitable for inverse kinematics. CADGene works well for simple simulations but is inappropriate to use with humanoids that need more advanced sensory input. Furthermore, the pressure sensors were found to suffer from inaccuracy when used without operation amplifiers.

Acknowledgment

The work on this master thesis was carried out at the university of Dortmund, Germany and at Chalmers University of Technology, Göteborg, Sweden. In Dortmund, Jens Ziegler at the Department of Computer Science was my thesis advisor. At Chalmers Peter Nordin was my advisor to begin with. Andrew Wallace commented on the work after Peter Nordin left Chalmers. The last stage of this thesis work was supervised by Mattias Wahde at the research group Adaptive Systems. I would also like to thank Jan Barnholt who created a virtual robot model (for the simulation environment SIGEL) of the physical robot ZORC. Finally I would like to thank Maria Baltussen and Thomas Krantz for proofreading the thesis.

Contents

Chapter 1

Introduction

1.1 Motivation

For industrial robots it is common to use inverse kinematics and model-based control. This has been studied in [14] for four-legged walking machines, and the results obtained sparked the question whether model-based control could be used successfully on bipedal robots.

In the near future, it is expected that humanoid bipedal robots will generate multiple new fields of commerce and help the civilian, military, and scientific communities in several ways. The development of robots such as ASIMO [9] has shown many of the possibilities of walking robots. Work on much less sophisticated robots such as ELVINA [19] has also demonstrated that that interesting bipedal behaviour could be studied with limited resources.

The simulation environment Sigel [18], designed to evolve code to control robot behaviour, was a project work by students at the ”Lehrstuhl für Systemanalyse”, University of Dortmund, Germany. Ziegler [20] was interested in testing the simulation system with real robots. The university also had a small bipedal robot, ZORC [20], that had ties to Chalmers and that appeared suitable for a student to do research on. This robot could already walk, although slowly.

1.2 Purpose

Three-dimensional simulation environments have evolved significantly in recent years. Today it is possible to simulate car crashes, aerodynamic flows and to calculate the optimum movements for industrial robots with great accuracy. Part of this thesis investigates if the two simulation environments SIGEL [17] and CADGene [5] are useful during the development process of walking robots.

Inverse kinematics and model-based control has been used for several years in industrial robotics. It can be used to calculate the best path in space that each joint and limb should follow from one position to another. Also it has been used in computer animation and the production of computer games in order to make limbs move realistically. Since it has been used successfully both for industrial robots and for designing lifelike animated creatures it seems that it may also be useful when trying to improve the walk of bipedal humanoid robots.

Mathematical software packages like Mathematica are today able to solve complex analytical equations. It is therefore examined if such software can be used for finding inverse kinematic solutions for the humanoid robot ZORC.

1.2.1 ZORC

The original version of the robot used in this thesis was designed by Wolff [19]. An identical copy, named ZORC, was assembled at the University of Dortmund and is shown in figure 1.1. The design was later improved by Ziegler [20] and Barnholt [3] who added sensor systems, most importantly the sensors for the rotation angles in the servo motors. Specific details are found in section 2.1.

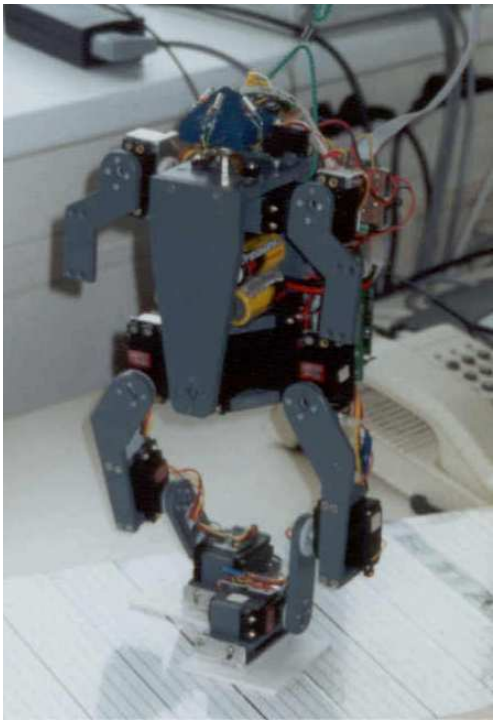

Figure 1.1: ZORC seen from the front.

1.3 Challenges

Balancing can be a problem even for humans. Making a robot walk, with less sophisticated senses than humans, while keeping its balance is problematic. The statically stable walk[1] is slow enough

[1] A robot is statically balanced when it does not move and no parts except its feet are touching the ground. A robot does a statically stable walk when from being statically balanced the robot will return to being statically balanced after experiencing a sufficiently small, nonzero, external force causing movement.

to disregard momentum that could otherwise make the robot fall.

ZORC is undergoing continuous development and the design has changed several times. The material that was used to build it, see section 2.1, is slightly soft and flexible. The position of servos and joints can therefore shift slightly. Also the wiring is a bit loose because of the design and how the EyeBot [7], the Micro-controller used to control ZORC, is fitted to the robot. These characteristics cause perturbations that cannot be eliminated. Instead the balance and motion algorithms must minimize their effect on the desired behaviour. Unfortunately many simulation environments do not have the ability to easily introduce random perturbations in their simulated systems. This is a common problem when moving from a virtual robot to a physical robot.

Chapter 2

Analysis

2.1 ZORC

The robot [20, 3] is mostly made from PVC[1]. It weighs two kilogrammes in total. Standing on its feet, as seen in figure 1.1, it measures 37 cm in height. The proportions are similar to those of a human. Each arm has two degrees of freedom and each leg has four. Adding them gives a total of twelve degrees of freedom. The feet have been designed to be simple but still similar to the human foot. Therefore each foot is slightly wider at the toes than at the heels, measuring 65 mm across the toe area, 80 mm in length and 60 mm across the heel.

Commercial mini servos from Hitec[2] are used for controlling ZORC's arm and leg movements. At the time the servos cost roughly 60 euros each. For each hip, knee and foot joint the HS-945MG servo capable of generating a torque of 0.78 Nm is used. For each shoulder and arm-joint the HS-225MG servo capable of 0.48 Nm torque is used. The servos are controlled by the EyeBot Micro-controller, which was constructed specifically for use with mobile robots. The EyeBot has multitasking capabilities although limited ones. It has eight digital input ports and eight digital output ports where two are used internally by the controller.

Rechargeable batteries are welded together to form a unit of six batteries are used to power ZORC. The welding was done to minimize battery movement, which would otherwise affect the robot greatly. This unit of three batteries on top and three below is placed inside the robot's torso to improve the weight distribution. Earlier the battery unit was fastened to the robots back, but this heavy backpack easily caused the robot to fall over. To facilitate easy swapping of the rechargeable battery pack, Velcro tape is glued to the inside of the torso and on the pack. Each foot has an on/off switch that switches on when the foot is experiencing a load greater than about one hundred grammes. These two simple pressure sensors are connected to digital in port 9 and 10. In order to sense inclination ZORC is equipped with 4 simple inclination sensors. They are mounted on the head in the shape of a pyramid with each sensor at a 45 degree angle against the horizontal ground plane. There are no joints in the robot's neck and therefore the sensors can tell when the upper body is leaning. These sensors are connected to the infrared port.

[1] Polyvinyl chloride

[2] Hitec RCD USA, Inc.

To increase the number of possible inputs of the EyeBot, ZORC uses a Mini SSC[3] II from Seetron[4]. To supply it with the correct and inverted power, ZORC uses the IC CD4001[5].

2.2 SIGEL

Sigel [17] is a simulation environment for mobile robots. It runs under Linux and was designed specifically to evolve control algorithms for walking robots using Genetic Programming. It does have the capability of using virtual sensors within the simulation environment.

2.3 Possible ways of control

ZORC is a bipedal humanoid robot capable of standing and walking to some degree. Experiments on how to make ZORC walk have previously been conducted using different approaches. One such approach was to manually program the robot according to a specific gait pattern. That is not an adaptive way of walking. Another approach used genetic programs found by repetitive evolution on the robot. Successful although slightly unstable gait patterns were found [20]. Further, evolving genetic programs in the simulation environment SIGEL as well as on the live robot was another approach [3]. These three different methods all produced somewhat stable behaviour although Ziegler, running the ZORC projects, believed that the behaviour could be further improved.

2.4 Stability control

When controlling the motion of bipeds it is necessary to solve at least two problems: stability control, the ability to maintain an upright body posture, and motion control, the ability to move in a desired direction at a desired speed.

2.4.1 Inverted pendulum

A simple method for balancing is the inverted pendulum approach. It works very well if there are few joints. For instance if the leg only has a hip joint and a knee joint then the upper body is easily kept upright. Problems occur when the number of joints are larger. Indeed the legs of ZORC resemble an inverted pendulum, see figure 2.1, although ZORC has two joints per leg, excluding the hips, and not one joint as the inverted pendulum.

[3] Serial servo controllers (SSC) enable data from a computer's serial port to control servos.
[4] Scott Edwards Electronics, Inc.
[5] An integrated circuit with four NOR-gates

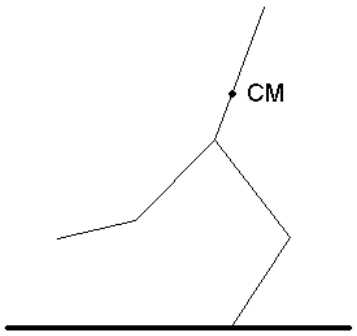

Figure 2.1: A humanoid robot can be approximated by an inverted pendulum.

2.4.2 Sensors and stability

Using force sensors, also called pressure sensors, is an economic way to help a robot balance [15]. They can be placed under the robot's feet so that they are the only contact points with the ground, seen later in figure 3.1. By measuring the force on each sensor, and by knowing the total mass of the supported robot, the robot's centre of mass projected on the surface that the robot is standing on can be calculated. ZORC already has inclination sensors fitted on its head with which it can roughly sense when it is falling over.

Other possibilities include using gyros, but they have a high cost compared to force sensors. Two other possible designs include using gravity: a laser pen on a string with a web-camera detecting where the light is shining thus detecting inclination; an air bubble in a liquid which will not be as prone to swinging or oscillating like a pendulum.

2.5 Motion control

Control algorithms for walking can be designed in various ways. There are four common approaches [2], which can be combined in hybrid forms.

2.5.1 Finite state control

Finite state controlled robots go through a sequence of moves. At the end of these moves the robot is supposed to have changed posture or advanced in a desired direction. While using this approach the trajectory of the joint angles are only known at certain points.

2.5.2 Modulated playback

By studying and tracking the limbs and joints of a moving animal or human, it is possible to produce an animated copy that can carry out the same moves as the real animal or human. This approach has been used on statically stable and dynamically stable walking robots since the

seventies. In the nineties it was also used for walking on uneven terrain. The robots used were WL-12RVI and WL-12RVII from Waseda University, Tokyo. Their feet were equipped with sensors that could detect forces and moments acting on them. By modifying the joint trajectories to match these forces and moments, uneven terrain with disturbances of up to 11 mm could be handled well [2]. Also the famous Honda P2, P3 and Asimo [9] robots all use a modified version of modulated playback [2]. They have added a dynamic model of the robot and a walking-pattern generator along with comparing the desired ZMP[6] with the actual measured ground reaction force [6].

Some problems do exist with this approach. Bipedal humanoid robots can naturally only push on the ground and not pull on it. They may also slip if the horizontal ground force is greater than the friction. Therefore the centre of pressure must not leave the support polygon of the foot with ground contact. It is also necessary to know the specifications of the joint trajectories. A great advantage with modulated playback is that it is easy to add new gaits or new tasks simply by recording the desired behaviour and storing it in the robot.

2.5.3 Mathematical synthesis

There are three recommended steps on how to develop a control system based on mathematical synthesis [2]. The first is to model the system including noise, disturbances and possible modelling errors. Second is to define the stability requirements and the desired performance. Third is to synthesise a controller and to prove that it is capable of performing according to the desired stability and performance requirements.

When modelling the system it helps to find linear relationships to simplify the equations of motion. Of course this is not always possible but for systems like rockets and aeroplanes it is a common approach. For bipedal robots things are more difficult. Bipedal robots are nonlinear and not easy to model mathematically. Therefore it is generally not recommended to try and design algorithms for bipedal robots with this approach. That said there are exceptions where the algorithms have been designed by using mathematical synthesis. One such exception is BIPER [2], built at the Tokyo University in 1984. To minimize any nonlinear effects its limbs only make small joint angle excursions and BIPER also has straight legs with joints only at the hips.

Advantages with this approach are the provability of the stability of the robot and the computability of specified performance requirements. The main drawback can be the complexity of finding the mathematical equations needed to specify the dynamics of the system.

2.5.4 Passive dynamics

Instead of developing algorithms from how a robot is built it is possible to design a robot after how it should behave, without any need for computing power. Numerous toys are built after this principle. A paper aeroplane that glides through the air and a toy walker, walking, or wobbling down a slope are two examples. Designing robots after the same principles has also been tried [2, 8]. Specifically, actuators have been used to increase the speed and range of passive walkers.

[6] Zero moment point

Actuators can also give the passive walkers the ability to walk uphill. Problems do arise when trying to make these dynamically stable walkers walk on uneven surfaces or walk by using different gait patterns. On uneven terrain they easily fall or stop [2]. Even so passive walkers have recently been made to navigate uneven terrain [8]

2.5.5 Physics based heuristics

As the name suggests, simple models approximate dynamic behaviour for use in the walking algorithm. The approach can eliminate the need for an internal dynamic model. A problem is that it often requires some insight and several iterations before the simple physics models can be adjusted sufficiently to control the walk.

2.6 Inverse kinematics and model-based control

Inverse kinematics has long been used to model industrial robots for various tasks. It is used to calculate the joint angles needed to reach a certain target. Inverse kinematics is based on forward kinematics, discussed in appendix A.

Pieper [16] showed that a six degree of freedom manipulator whose last three joints intersect always has at least one inverse kinematics solution. This assumes that the target is in the mechanisms workspace i.e. the target is within physical reach. There are several numerical and analytical methods for solving inverse kinematics problems.

2.6.1 Solving the inverse kinematic equations

The Denavit-Hartenberg or DH-approach, explained in appendix A.1, relies on reducing the inverse kinematics problem into subproblems, focusing on one joint at a time. DH uses a four by four transformation matrix for each joint to describe the rotation and possible translation from one side of the joint to the other. When there are several links in a chain then the individual matrices for each link are simply multiplied together to describe the relationship between the start and end of the chain of links. It is therefore possible to use DH even with complex bodies with many joints. A drawback is that it will not work for elastic links and it is therefore limited to rigid bodies.

Dialytical elimination, like the DH-approach, is a very general method. It works by reducing the problem to a system of polynomial equations. These equations are then solved analytically by use of algebraic geometry and elimination theory. For that reason it seems to be well suited for use with software such as Maple or Mathematica. On the other hand the polynomial equations have normally not been simplified by using the kinematic properties of the robot and may therefore be too complex.

As for numerical methods it is possible to adapt Newton's root finding technique. By using forward kinematics, the joint transfer matrices and the inverse of the solution can be found after some iterations. Care must be taken to ensure that the algorithm remains stable and will converge [13].

Chapter 3

Method

3.1 Balancing with ZORC

The chosen way for calculating the centre of mass is using force sensors as presented in [15]. The placement of the sensors is shown below.

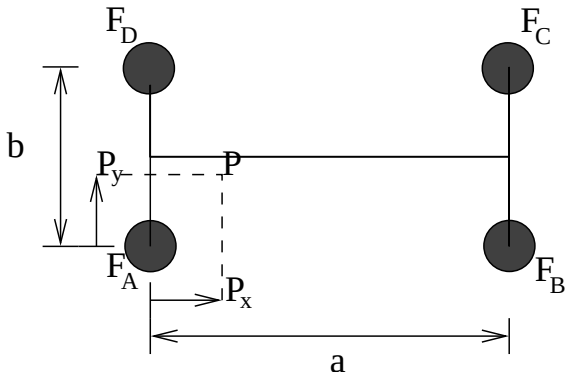

Figure 3.1: A foot design with four circular force sensors.

Using this type of foot design the centre of mass can be calculated for a sufficently slowly moving robot using the following equations:

$$P_x = \frac{F_B + F_C}{W} a \tag{3.1}$$

$$P_x = \left(1 - \frac{F_A + F_D}{W}\right) a \tag{3.2}$$

$$P_y = \frac{F_D + F_C}{W} b \tag{3.3}$$

$$P_y = \left(1 - \frac{F_A + F_B}{W}\right) b \tag{3.4}$$

Here F_A, F_B, F_C and F_D represent the force on each supporting corner of the foot and W is the total force on the foot. Since the equations measure reaction force and not inertia they actually calculate the centre of pressure rather than the centre of mass but for sufficiently slow speeds the two will be the same. There is one redundant equation for P_x and one for P_y. This redundancy can be used to increase robustness and lower error influence when calculating the centre of mass, P. If a biped is standing on one foot and P is found to be outside the supporting area of the foot then the robot will fall over. If on the other hand it is standing on both feet, then the stable area for P is larger as seen in figure 3.2. As long as the speed of the robot is sufficiently low for the momentum to be negligible and the projection of the centre of mass lies within the polygon P then the robot will be statically stable.

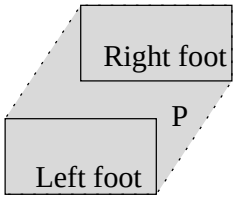

Figure 3.2: Stability area for the robot's centre of mass when both feet have ground contact.

This means, disregarding momentum, that when the biped is taking a step forward and has one foot in the air it has to put the elevated foot down before the centre of mass, projected on the ground, may leave the area of the foot with ground contact.

3.1.1 Force potentiometers

ZORC's feet are rectangular, as those proposed in [15]. Thereby, force sensors are placed underneath the feet. A thin, small and circular cut piece of plastic is placed beneath the sensor to better transfer pressure from the ground to the sensor.

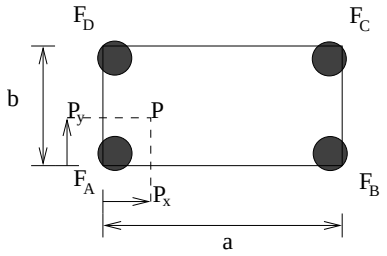

Figure 3.3: ZORC's rectangular foot design with added circular pressure sensors.

FSR^{TM}-force potentiometers from IEE[1] as seen in figure 3.4 were chosen to be used with ZORC for this thesis. The main reason for this is that they are cheap and sufficiently accurate

[1] International Electronics Engineering

according to the data sheets. The resistance is proportional to e^{-F}, where F is the force applied to the sensor surface. Specifically the FSR^{TM}-149NS satisfies the size and pressure sensitivity requirements for ZORC.

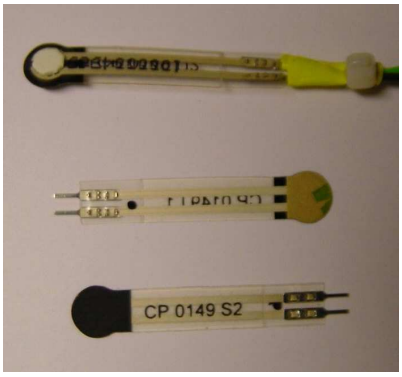

Figure 3.4: The FSR^{TM}-149NS sensor used for sensing pressure.

The potentiometers are connected to an analogue-to-digital converter, ADC, which in turn is interfaced with ZORC's on-board micro controller via its in and out ports. Together they form what will be referred to as the COM[2]-sensor. The connection between the eight force potentiometers and the circuit with the ADC are done at points y_1 through y_8, seen in figure 3.5. This circuit uses voltage division between R and R_{poti}, where i is the channel number from 1 to 8 and R_{poti} is the resistance of force potentiometer i. It is supplied with a voltage of $V_{cc} = 5V$. The voltage at input channel i of the ADC is

$$V_i = V_{cc}(1 - R/(R + R_{poti})) \tag{3.5}$$

The weight of ZORC is 2 kg. According to the developers of ASIMO [9], the ground reaction force will not exceed 1.4 times the robot's mass. If this is correct also for ZORC then the force on a single sensor should not exceed 2.8 kg. Since there are four sensors on each foot, each sensor will experience an approximate load of 250 g when ZORC is standing still. The sensors also need to be able to measure smaller forces in order to detect the centre of mass. As an estimate for the design process the sensors must be able to measure forces in the range of 50 g to 3 kg.

The sensor circuit is powered by 5 V. To measure forces in the desired range it is possible to use $R = 40k\Omega$. As seen in table 3.1 a 50 g load would give a resistance for the potentiometer of 65 kΩ. Voltage division over the ADC and the potentiometer:

$$V_i = 5(1 - 40/(40 + 65))V = 3.1V \tag{3.6}$$

This is the same as an eight bit digital voltage value of 157 on the ADC's digital port.

$$V_{\text{digital}} = 255(V_i/V_{cc}) \tag{3.7}$$

[2]Centre of mass

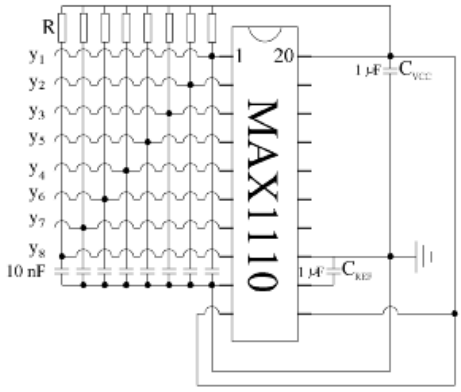

Figure 3.5: Schematic of the COM sensor (ADC and wiring). R = 40kΩ.

A load of 500 g and 22 kΩ resistance for the force potentiometer, see table 3.1, will result in $V_i = 2.0$ V or a digital reading of 100. Therefore 40 kΩ is used for testing if the force sensors will function with ZORC. The sensor readings must be calibrated. The calibration is done using weights ranging from 50 g to 500 g as is seen in the following table.

Weight (g)	R_A (kΩ)
500	22
125	34
50	65
0	>1000

Table 3.1: R_A is the resistance of one force sensor measured over a short duration with an ohmmeter when a weight is put on the sensor.

The sensors are fitted underneath a rectangular plate similar in size to ZORC's feet. Then the calibration weights are applied on top of the plate and the resistance of each sensor is measured using the ADC and the sensor circuit. Seen from above the sensors are connected to the ADC pins as seen in figure 3.6. The readings in table 3.1 fluctuated with about 10 percent around the values shown. To increase the stability of the readings it may be necessary to use operation amplifiers, as suggested in the data-sheet for the sensor [10]. However, this was not done because of the near linear relationship between resistance and force above 1 Newton, as seen in figure 3.7. Further, if the sensors are not precise enough to accurately measure the centre of mass they will still be able to measure if there is ground contact.

ZORC's backside

Right Foot		Left Foot	
channel 6	channel 7	channel 1	channel 2
channel 4	channel 5	channel 3	channel 0

Figure 3.6: Pressure sensor placement with connections to specific channels on the ADC.

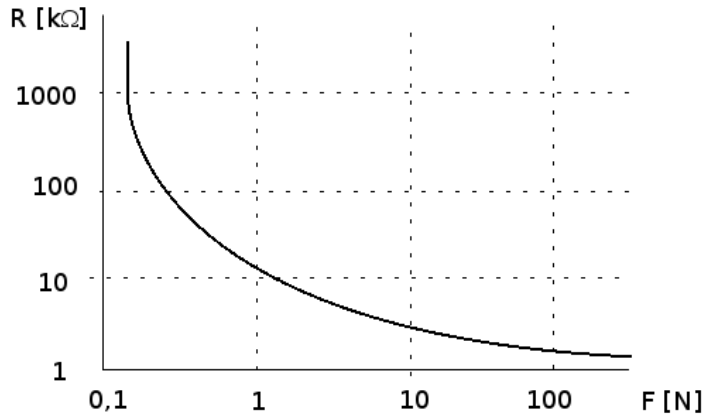

Figure 3.7: Force sensor resistance in relation to applied force.

3.2 Walking

A bipedal robot can be modelled as two legs carrying a movable weight, similar to figure 2.1. Such a model makes it easier to focus on the essentials of walking. Section 2.6 showed that inverse kinematics can be good for keeping an internal representation of how to move the robot. The matrices describing the kinematics of the robot contain big and complex expressions. Therefore Mathematica is used to help simplify the equations. The calculations can be found on-line in a mathematica code file [12]. The specific solution for ZORC to this four by four matrix equation can be seen in section 3.2.1. The equations have been tested [12] in MATLAB.

While walking, and keeping the upper body erect, one challenge is knowing precisely how the legs shall move. Therefore effort has been made to find an inverse kinematics solution mainly for ZORC's legs. There are a number of physical restrictions caused by the design of ZORC in terms of maximum angles for the servos. These restrictions will be handled in the MATLAB simulations. Also, there are aesthetic restrictions that may be considered. For instance if the robot is supposed to simulate human movement then the knee joint should not be able to bend forward.

3.2.1 Leg model

A virtual model of one of ZORC's legs with three joints and four degrees of freedom is shown in the figure below. This model is used for finding an inverse kinematics solution that is useful when programming ZORC.

When finding the inverse kinematics solution the forward kinematics solution is assumed to be known. It can be written in matrix form, with joint angles specified in modified Craig Denavit Hartenberg form, as the following:

$$\mathbf{A_i} = \begin{pmatrix} \cos(\psi_i) & -\sin(\psi_i) & 0 & u_i \\ \cos(\theta_i)\sin(\psi_i) & \cos(\theta_i)\cos(\psi_i) & -\sin(\theta_i) & -\sin(\theta_i)w_i \\ \sin(\theta_i)\sin(\psi_i) & \sin(\theta_i)\cos(\psi_i) & \cos(\theta_i) & \cos(\theta_i)w_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

$$T_{destination} = A_1 A_2 A_3 \dots A_n \tag{3.9}$$

where also

$$T_{destination} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

Here r_{11} to r_{33} make up the desired rotation relative to the starting direction and (p_x, p_y, p_z) is the desired destination position. These 12 elements give 12 possible equations when knowing the destination position and direction of the linkage. There are two cases that need to be considered. One if ψ_2 is zero and the other if ψ_2 is nonzero, where ψ_2 describes the sideways angle of the hip

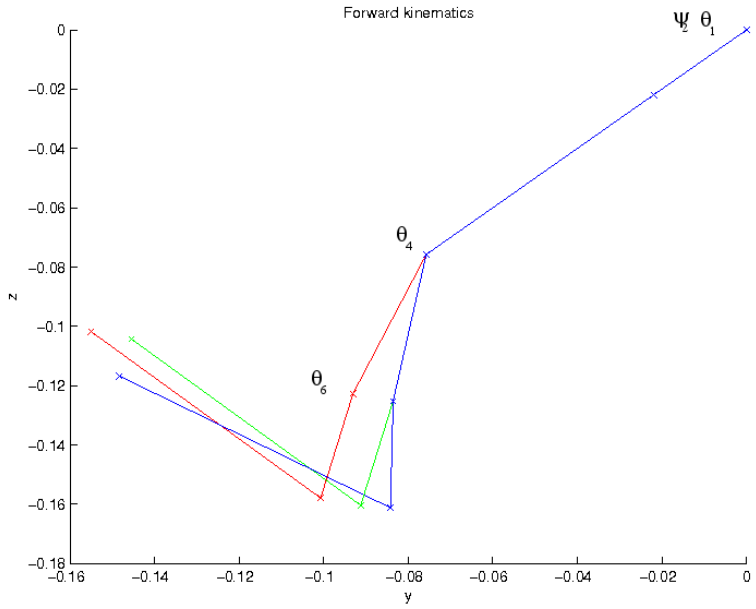

Figure 3.8: The plot shows a leg linkage while varying two joint angles. The green line shows a change for the ankle joint and the red line shows a change for the knee joint. The blue line describes an initial leg posture.

joint. All other joints have the same forward direction, and therefore if ψ_2 is zero the problem is two-dimensional. Thus,

$$\theta_1 - \theta_4 - \theta_6 = -\arccos(r_{21}) \tag{3.11}$$

$$\theta_{1+} = \arcsin\left(\left(c_1c_2(4(c_2^2+c_3^2)+c_1^2-4L_4^2)+ \sqrt{-c_1^2c_3^2(c_1^4+16(c_2^2+c_3^2-L_4^2)^2-8c_1^2(c_2^2+c_3^2+L_4^2))}\right)\Big/\left(4c_1^2(c_2^2+c_3^2)\right)\right) \tag{3.12}$$

$$\theta_{1-} = \arcsin\left(\left(c_1c_2(4(c_2^2+c_3^2)+c_1^2-4L_4^2)+ \sqrt{-c_1^2c_3^2(c_1^4+16(c_2^2+c_3^2-L_4^2)^2-8c_1^2(c_2^2+c_3^2+L_4^2))}\right)\Big/\left(4c_1^2(c_2^2+c_3^2)\right)\right) \tag{3.13}$$

where

$$c_1 = r_{21}L_6 + r_{23}(L_{6z}+L_{7z}) - p_y \tag{3.14}$$

$$c_2 = r_{23}L_6 - r_{21}(L_{6z}+L_{7z}) - p_z \tag{3.15}$$

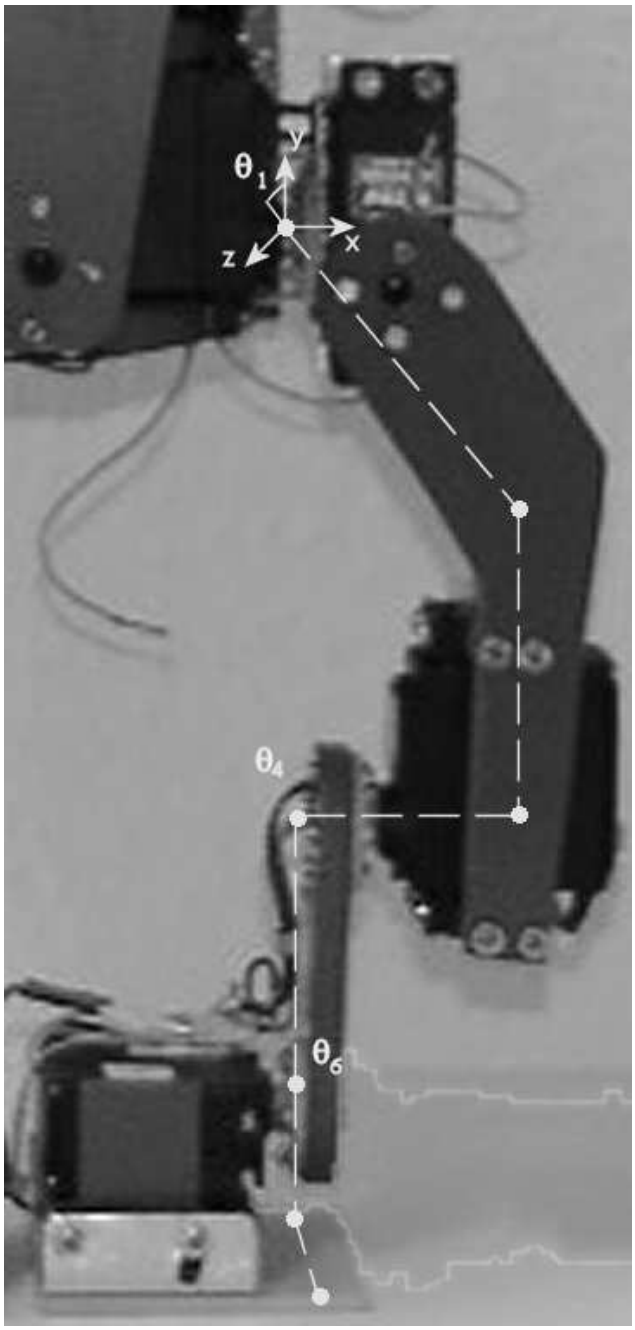

Figure 3.9: Joints used for ZORC's left leg.

$$c_3 = \sqrt{2L_1 + 2L_2} \tag{3.16}$$

$$c_4 = \sqrt{-c_1^2 c_3^2 (c_1^4 + 16(c_2^2 + c_3^2 - L_4^2)^2 - 8c_1^2(c_2^2 + c_3^2 - L_4^2))} \tag{3.17}$$

$$\theta_{4-} = -\arcsin\left(\frac{-(\sqrt{2}L_1 + 2L_2)\sin\theta_1 - 2p_z - 2L_{6z}r_{21} - 2L_{7z}r_{21} + 2L_6 r_{23}}{2L_4}\right) + \theta_1 \tag{3.18}$$

$$\theta_{4+} = -\arcsin\left(\frac{+(\sqrt{2}L_1 + 2L_2)\sin\theta_1 - 2p_z - 2L_{6z}r_{21} - 2L_{7z}r_{21} + 2L_6 r_{23}}{2L_4}\right) + \theta_1 \tag{3.19}$$

When ψ_2 is nonzero the problem is three-dimensional. Thus,

$$\psi_2 = \arccos(r_{12}) \tag{3.20}$$

$$\theta_4 + \theta_6 = \arctan(-r_{13}/r_{11}) \tag{3.21}$$

$$\theta_1 = \arctan(r_{32}/r_{22}) \tag{3.22}$$

This results in:

$$\theta_{4-} = -\arccos\left(\frac{\sqrt{2}L_1 + 2(-p_x + L_6 r_{11} - L_3 r_{12} + (L_{6z} + L_{7z})r_{13} + L_2\sqrt{r_{22}^2 + r_{32}^2})}{-2L_4\sqrt{r_{22}^2 + r_{32}^2}}\right) \tag{3.23}$$

3.2.2 Robot model and numerical solution in MATLAB

In the case of a more complex robot model it can be difficult to solve the inverse kinematic equations analytically, even when using software like Mathematica. Also if a modification of the robot model is made then the analytical solution will have to be reevaluated, possibly with a very different result compared to the previous solution. Therefore a numerical approach using MATLAB is also tried.

The program, developed for this thesis and found in [12], accepts any robot model file that is specified in Denavit Hartenberg standard form or modified Craig form, and plots the model as seen in figure 3.10. The centre of mass is calculated from the mass of each link. The centre of mass for the entire model is calculated by summing the centre of mass for each individual link and dividing by the total mass of the model. The centre of mass is then used to calculate how to keep the model statically stable.

MATLAB has many built in commands for solving equations numerically. One such is fgoalattain and another is fsolve. The first attempts to find function inputs that produce specific outputs. The other tries to minimize the output of a given set of functions. With the aid of these commands, numerical solutions to the inverse kinematic equation are found even when using constraints. Such constraints include joint angles not varying more than specified, and stability criteria that maintain the centre of mass in a stable area.

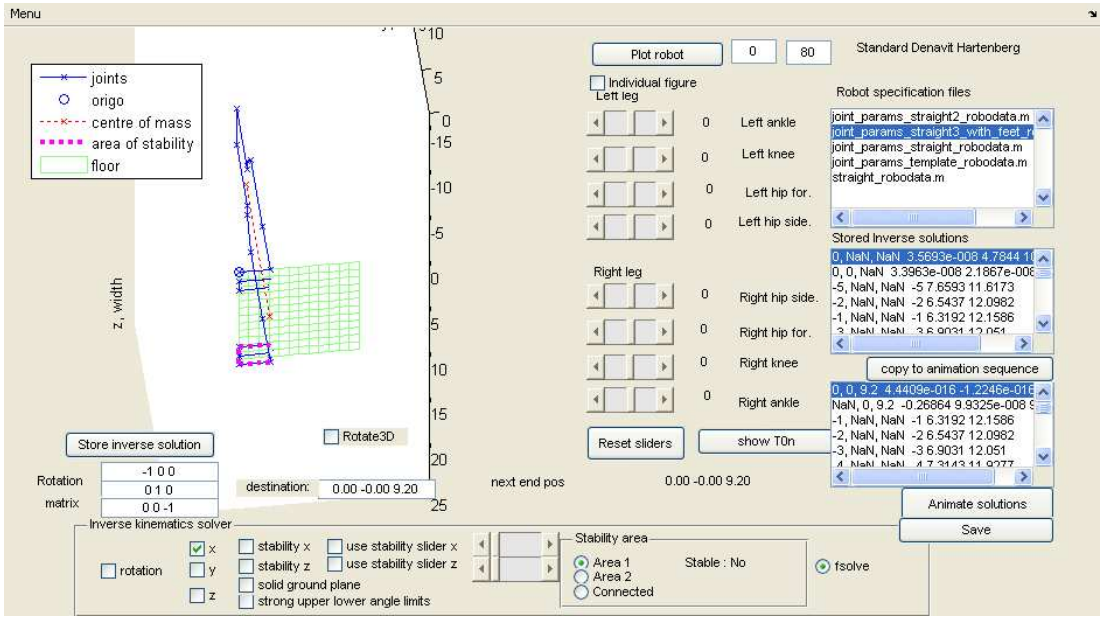

Figure 3.10: MATLAB robot simulator program.

The numerical approach to inverse kinematics, like the analytical approach, uses the desired destination position and rotation to solve for the joint angles, as seen in matrix equation 3.9. If it is only the position part of the matrix equation that is of interest then the matrix equation will produce only three equations and the system will be under-determined. This does not matter as long as at least one solution is found that fit the constraints. The numerical approach used in MATLAB will find a solution close to an initial guess. That is if the last position or inverse solution is used as a starting guess then a solution close to that guess will be found if one exists. With this approach the solutions will be close to each other when possible and therefore be good for animating a walk when they are viewed in succession from the starting position to the goal position.

The exponential function:

$$f = -1 + 10^a \tag{3.24}$$

where a is the sum of the absolute values of how much each angle has exceeded its constraint, is used for the angle constraints of the joints. Minimizing this function while finding joint angles that lead to the destination position will keep the joint angles from greatly exceeding the angle constraints. The other constraints were added in similar manner, as can be seen in [12].

No code from external sources is used, except a function for finding the shortest distance between a point and a polygon in space used for stability calculations. The function was created by Michael Yoshpe.

3.3 Balancing with CADGene

In order to test the possibility for a robot such as ZORC to stand up if it falls, a simulation experiment is conducted. This because of speed and because of the need to minimize damage to ZORC while testing. The environment chosen was CADGene because it was interesting, according to Nordin [19, 20], to investigate the possibilities of CADGene. The model itself was built in a modified version of the 3D model drawing program Aztec. The necessary modifications for Aztec to work with CADGene had already been implemented by the CADGene developers [5]. The model designed for CADGene during this thesis work is similar to ZORC. The design can be seen in the figure below. The Aztec and Creml code for this model is found in [12]. CADGene and Aztec do not allow the rotation of limbs and joints after the model has been created. Therefore the model is lying down, flat on its back, as a starting position. Because of limitations of CADGene and Aztec this model of ZORC is not as exact as Barnholt's model [3] used with the SIGEL system.

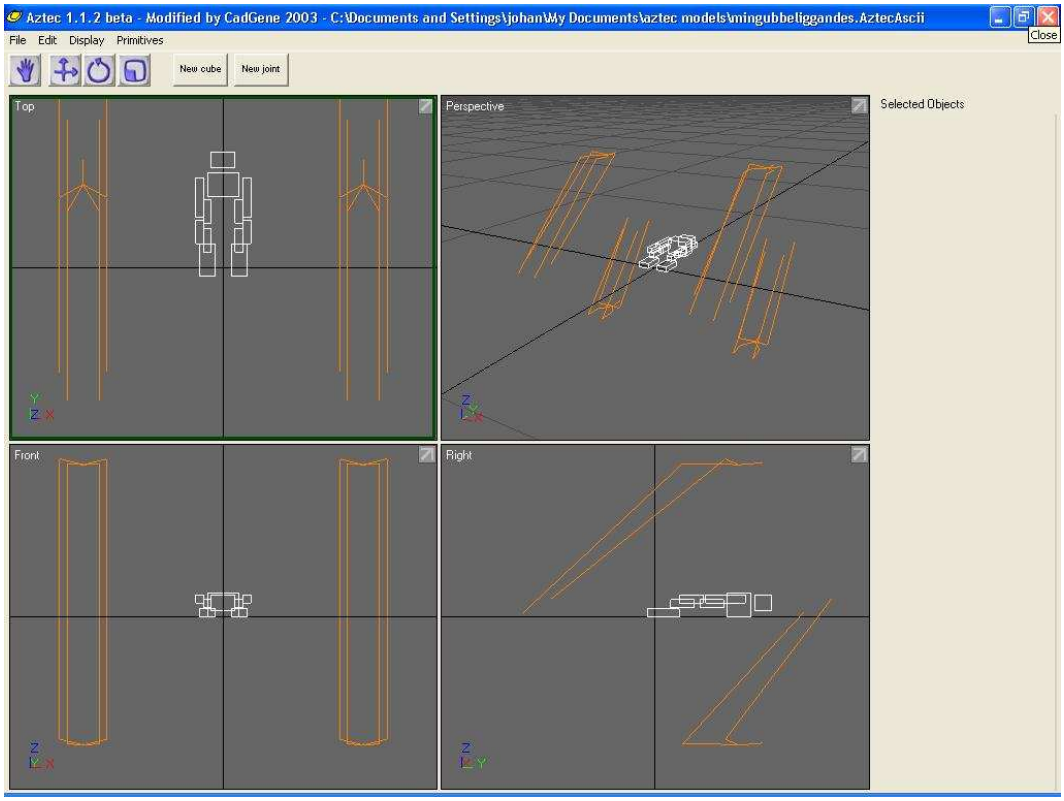

Figure 3.11: Robot design made with Aztec and used in the CADGene experiments.

3.3.1 Fitness functions in CADGene

It is often recommended to keep fitness functions simple. It makes it easier to understand the results and the function will also likely better describe the desired behaviour. In the experiments with CADGene we want the robot to be able to stand up from a lying down position. What we want is therefore to get the highest point of the robot, the head, as far from the ground as possible. Therefore the distance from the head to the ground, at the end of each genetic program being run, was chosen as a fitness function. Other functions could easily be tested but this first choice produced good enough results. The fitness function file is found in [12].

Chapter 4

Results

4.1 Centre of mass sensor circuit

ZORC is seen from behind in figure 4.1 with pressure sensors attached under the feet.

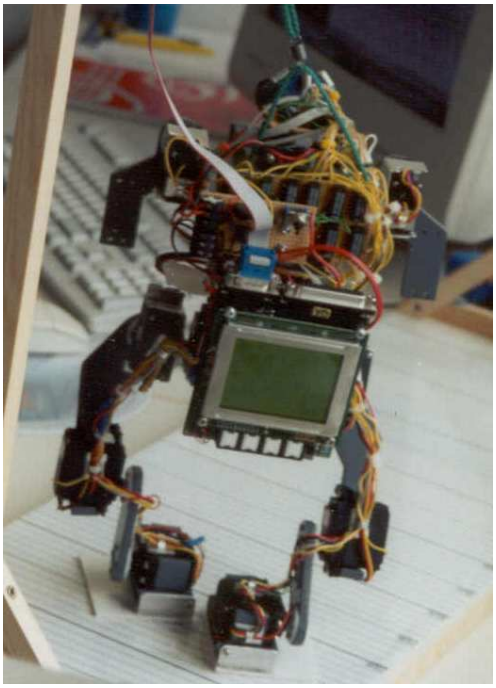

Figure 4.1: ZORC carrying the EyeBot Micro-controller.

The COM-sensor, see figure 3.5, interfaced well with the EyeBot. It monitored the pressure sensors and delivered the readings to the controller as expected, see table 4.1.

The obtained readings show that the force sensors do not give accurate results. For example a load of 50 g on one sensor and 125 g on a different sensor can give the same values. The results

Applied weight on each foot	0 g	50 g	125 g
ADC channel 0	255	155	100
ADC channel 1	255	165	111
ADC channel 2	255	178	140
ADC channel 3	255	155	113
ADC channel 4	255	165	112
ADC channel 5	255	145	97
ADC channel 6	255	155	120
ADC channel 7	255	168	115

Table 4.1: The readings from the ADC channels' eight force sensors when weights are applied on the sensors.

vary and do not stabilize around one value for a specific load. The readings from the pressure sensors, meant to be used to calculate the centre of mass, fluctuated too much.

4.2 Analytical solution and simulation of ZORC's leg using inverse kinematics

The inverse kinematic equations 3.11 to 3.23, were tested successfully in MATLAB. Because of symmetry only the left leg of ZORC was used in the simulation. The MATLAB test showed that the equations for the angles did move the leg to the desired destination position. Forward kinematics was used to verify the result. The simulated leg can be seen in figure 3.8 on page 15.

The code for the MATLAB program is found on-line in [12].

4.3 Robot visualizer in MATLAB

The robot visualizer program developed for MATLAB in this project was able to find numerical inverse kinematics solutions to a robot specification file that describes a robot similar to ZORC. The specification file, named `joint_params_straight3_with_feet_robodata.m`, can be found in [12]. The starting position for this specification file, where all angles in the program are set to zero, is seen in figure 3.10 on page 18.

As can be seen in figure 4.2 the simulated robot can take a step forward without letting the centre of gravity leave the area of stability. This includes, see frames 10 to 20, that the robot model can shift the centre of mass from the right foot to the left without letting the centre of mass leave the stable area between the two feet placed on the ground. This means that the robot model can take a step without falling.

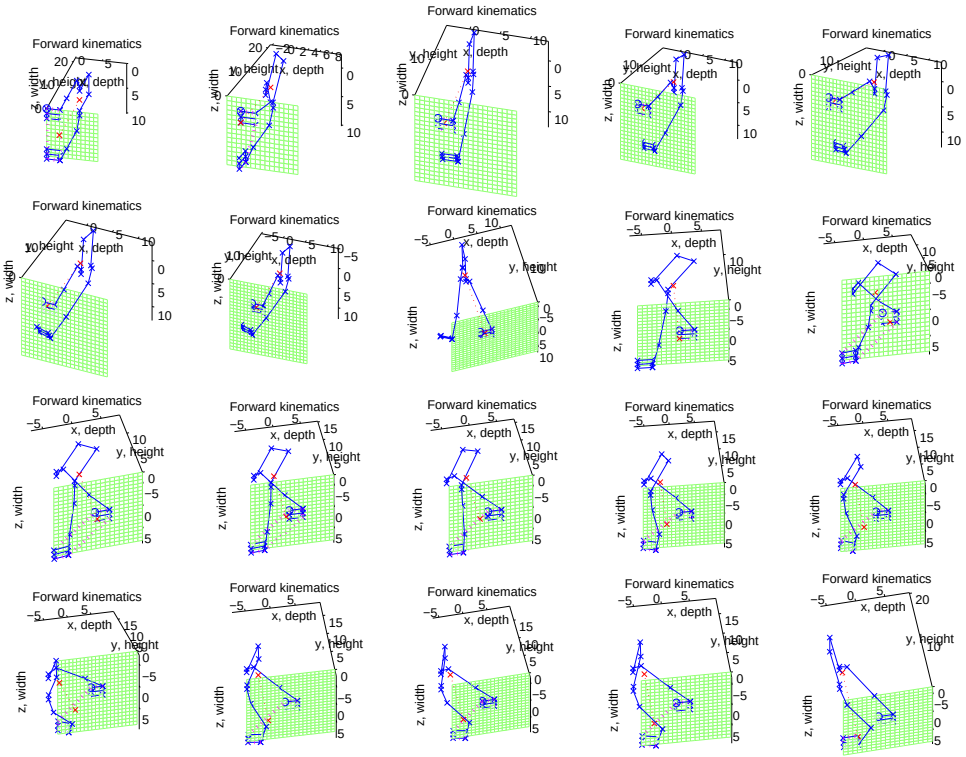

Figure 4.2: The starting position (frame 1) with support first on both feet, then shifting the centre of mass to the right foot followed by taking a step forward (frames 3-9) and finally shifting the balance to the left foot (frames 10-20).

4.4 Simulation in Sigel

The simulations appeared promising at first but the problem of adding virtual foot sensors to the code and bypassing the generation of genetic programs were not overcome. Sigel could not be used with inverse kinematics even though considerable effort was put into understanding, modifying and compiling the existing Sigel source code.

4.5 Simulation in CADGene of ZORC rising

As seen in the figure below the robot uses its arms to try to rise. The arms are not completely human-proportional and therefore it almost looks like it is using crutches in the picture. This was the only way found to get the robot to stand up in CADGene while trying to retain the main design of ZORC. An animated movie from the CADGene experiment can be found in [12]. As can be seen in the figure below the fitness value went from 0.6 to 2.1 where it stabilized as the model repeatedly reached an upright posture.

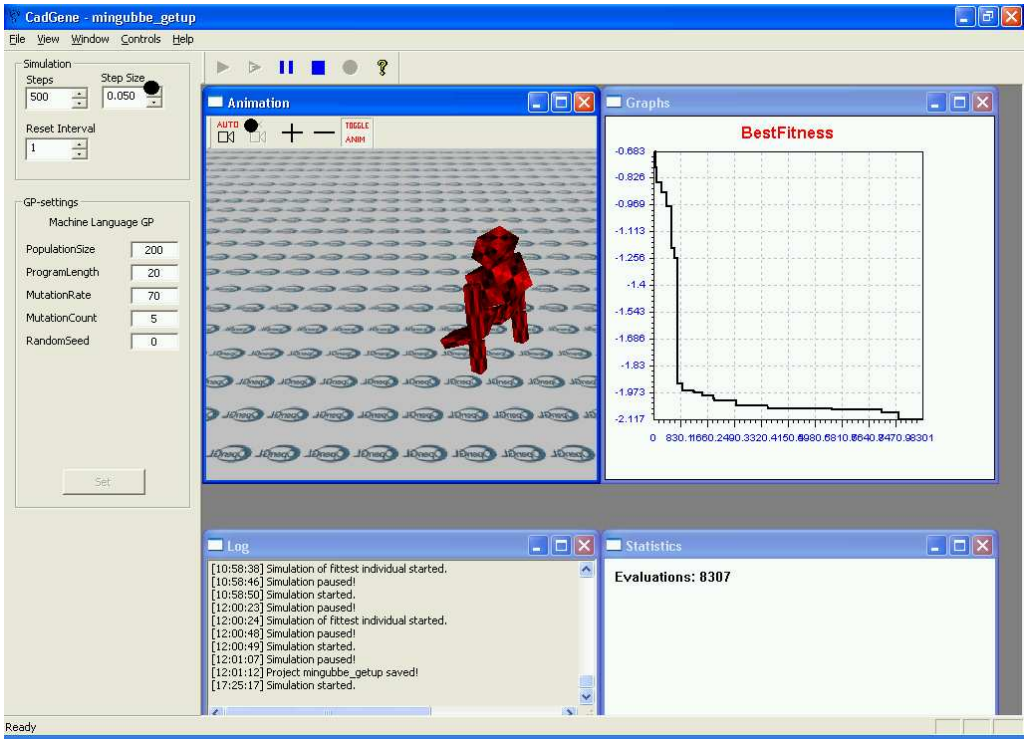

Figure 4.3: Robot attempting to stand up in a CADGene experiment.

Chapter 5

Discussion

5.1 ZORC

Sensors were added to ZORC's feet as part of this project. However, during the time spent in Dortmund with access to ZORC, there was at least one other project [3] running on ZORC and since the pressure sensors interfered with that project they could not remain on ZORC.

Even though ZORC lacks some joints to fully imitate human-like biped walking, it is a good platform for learning about humanoid walking and running experiments on.

5.1.1 Sensors

Calibrating the sensors was challenging. This is most likely due to the fact that the resistance decreases in an inverse exponential manner related to the pressure upon them. Also a better method than using weights for applying a known pressure on the sensors should be considered. Also, applying greater pressure than used for the calibrations should be tried. Reasons for not using operation amplifiers were that they would need extra power, cables and cause a slight increase in weight. They would also add complexity to the COM-sensor, which could become more error prone, if only because of the soldering done by hand. Still, seeing that the COM-sensor did not produce accurate results, it would be interesting to investigate the change in accuracy if operation amplifiers are used to make the response more linear.

5.1.2 Analytical solution to inverse kinematics

The inverse kinematics solution from section 3.2.1 was only valid for one leg and not the whole robot. That was not very useful. Some effort was spent on solving the equations for more than just one leg but due to adaptability concerns and time limitations this was abandoned. Instead effort was made to investigate a possible numerical approach that appeared more adaptable than the analytical approach. The analytical approach only works well with the robot model that the inverse kinematics solution is based on. If needed, an analytical solution can always be developed when it is clear that no further modifications will be made on the robot, until then a numerical approach is probably more rewarding.

5.1.3 Robot visualizer in MATLAB

There is at least one significant advantage to using a numerical approach in MATLAB over the analytical approach tried with Mathematica. If changes are made to the robot model, as in this project, then the analytical solution has to be corrected which is time consuming and difficult. The numerical approach however needs no change in the algorithm, making it easy to change the robot model and still solve the inverse kinematic equations.

The visualization program is also excellent for checking where the centre of mass will be for different postures. As such and as a visualization aid it can help identify where to add joints or limbs in order to get a desired behaviour from a robot under development. Although any number of joints and limbs can be added to a joint specification file and then viewed in the visualizer, there is a limit of eight variable angles that the program can use. If more variable joint angles are needed they should be added to the GUI.

5.2 Sigel

The Sigel project is definitely well suited for robot models programmed by genetic programming. It is not suited for incorporating inverse kinematics or adding extra functionality. A lot of strange errors cropped up while trying to modify and compile the Sigel source code. Also Sigel is not well documented which makes it difficult to understand exactly where and how to add code.

5.3 CADGene

Two drawbacks worth noting arose while using CADGene and Aztec. It was not possible to rotate a model around one common point in space. Instead of rotating every object around a single point the version of Aztec that had been modified for CADGene rotated each object around its individual centre, which was a fixed position that had been decided by Aztec. Rotating a model therefore separated its parts from each other due to the rotation. Thus to make a robot lie down if it previously stood up would be very time consuming since each individual object would have to be moved to the correct location after rotation. A greater problem is that CADGene does not support sensor input in the GP code. Therefore it is difficult to find a realistic way for models in CADGene to be able to keep their balance.

Additionally some challenges with the CADGene implementation were encountered. When exporting files from the modified Aztec environment, they did not contain specifications for a ground plane on which the robot could stand. Such a plane had to be manually added to each creml-file, [5]. Therefore each CADGene simulation has one Aztec-generated creml-file and one manually patched creml-file, which are the ones used for the CADGene simulations. Even the gravity in the creml-file needed to be modified from a value of -10 to -2 for the simulations to work at all.

Because of the lack of sensor functionality in CADGene the behaviour described in the results are considered satisfactory. Due to this lack of sensor capabilities the experiments with making the robot stand up and keeping its balance were not developed further.

Bibliography

[1] Angeles, J., *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*, Hong Kong, Springer, 2003

[2] Bar-Cohen, Y. and Breazeal, C., *Biologically Inspired Intelligent Robots*, SPIE Press, 2003

[3] Barnholt, J., *Evolution von Laufalgorithmen für die humanoide Roboterarchitektur 'ZORC' mittels Genetischer Programmierung*, University of Dortmund, 2002

[4] Bekey, G.A., *Autonomous Robots, From Biological Inspiration to Implementation and Control*, Massachusets Institute of Technology, MIT Press, 2005

[5] Blomqvist, L., Cederman D., Karlsson F. et al, *CADGene, Rapport - D3 projekt nr 5, 2002/03*, Civilingenjörsprogrammet för Datateknik, Chalmers Tekniska Högskola, Institutionen för datorteknik, Göteborg, 2003

[6] Bräunl, T., *Embedded Robotics, Mobile Robot Design and Applications with Embedded Systems*, Springer-Verlag, 2003

[7] Bräunl, T., The Univ. of Western Australia, EyeBot, *http://www.ee.uwa.edu.au/~braunl/eyebot/* (last accessed June, 2005)

[8] Byl, K. and Tedrake, R., *Stability of passive dynamic walking on uneven terrain*, Proceedings of Dynamic Walking, May 2006

[9] Honda ASIMO Homepage, Advanced Step in Innovative Mobility, *http://world.honda.com/ASIMO/* (last accessed May, 2008)

[10] International Electronics Engineering, *http://www.iee.lu* (last accessed May, 2008)

[11] Kelly, A.J., *Essential Kinematics for Autonomous Vehicles*, Carnegie Mellon University, 1994

[12] Kristenson, J., Program files for the thesis, *http://www.kristenson.se/chalmers/mscthesis* (last accessed May, 2008)

[13] Kurfess, T.R., *Robotics and Automation Handbook*, CRC Press, 2005

[14] Mehren, C., *Kinematik und Dynamik einer vierbeinigen Gehmaschine*, Ph.D. Thesis, Gerhard-Mercator-Universität - GH Duisburg, VDI Verlag, Reihe 8, 1995

[15] Palacin, J., Donaire, O., Roca, J. and Marco, S., *Static walker foot design and implementation*, 4th International Conference on Climbing and Walking Robots, 2001

[16] Pieper, D. and Roth, B., *The Kinematics of Manipulators Under Computer Control*, Ph.D. Thesis, Stanford University, 1968

[17] Sigel project page, PG-368, *http://sigel.sourceforge.net* (last accessed May 2008)

[18] Sigel development page, *http://sourceforge.net/projects/sigel* (last accessed May, 2008)

[19] Wolff, K. and Nordin, P., *Evolution of Efficient Gait with an Autonomous Biped Robot Using Visual Feedback*, Genetic and Evolutionary Computation Conference, 2001

[20] Ziegler, J., Wolff, K., Nordin, P., and Banzhaf, W., *Constructing a Small Humanoid Walking Robot as a Platform for the Genetic Evolution of Walking*, Proceedings of the 5th International Heinz Nixdorf Symposium, 2001

Appendix A

Kinematics

To understand inverse kinematics one must first have basic knowledge of forward kinematics. Both describe how a set of joints and links relate to a certain goal, a position and direction, in space. Forward kinematics can show if a goal has been reached and inverse kinematics can be used to determine how the joints should bend in order to reach that goal. In forward kinematics the relative angles and distances between consecutive joints are known.

A.1 Denavit Hartenberg and homogeneous transformation matrices

The Denavit-Hartenberg method [1] and the modified Craig D-H method [11, 13] are widely used in robotics, especially for robots with arms. In order to calculate the position of each joint with origin:

$$\mathbf{o} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{A.1}$$

the following matrix operators are defined:

$$\mathbf{Trans}(\mathbf{u}, \mathbf{v}, \mathbf{w}) = \begin{pmatrix} 1 & 0 & 0 & u \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & w \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.2}$$

$$\mathbf{Rotx}(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.3}$$

$$\mathbf{Roty}(\phi) = \begin{pmatrix} \cos(\phi) & 0 & \sin(\phi) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.4}$$

$$\mathbf{Rotz}(\psi) = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 & 0 \\ \sin(\psi) & \cos(\psi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.5}$$

These operators are commonly used to describe the forward kinematics of the system [4, 13]. The transformation matrices from one frame to the next differ depending on whether the standard Denavit-Hartenberg form,

$$\mathrm{A_i} = \mathrm{T^i_{i-1}} = \mathrm{Rotz}(\theta_\mathrm{i})\mathrm{Trans}(0, 0, \mathrm{d_i})\mathrm{Trans}(\mathrm{a_i}, 0, 0)\mathrm{Rotx}(\alpha_\mathrm{i}) \tag{A.6}$$

$$\mathbf{A_i} = \begin{pmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.7}$$

or the modified-Craig form, is used:

$$\mathrm{A_i} = \mathrm{T^i_{i-1}} = \mathrm{Rotx}(\theta_\mathrm{i})\mathrm{Trans}(\mathrm{u_i}, 0, 0)\mathrm{Rotz}(\psi_\mathrm{i})\mathrm{Trans}(0, 0, \mathrm{w_i}) \tag{A.8}$$

$$\mathbf{A_i} = \begin{pmatrix} \cos(\psi_i) & -\sin(\psi_i) & 0 & u_i \\ \cos(\theta_i)\sin(\psi_i) & \cos(\theta_i)\cos(\psi_i) & -\sin(\theta_i) & -\sin(\theta_i)w_i \\ \sin(\theta_i)\sin(\psi_i) & \sin(\theta_i)\cos(\psi_i) & \cos(\theta_i) & \cos(\theta_i)w_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.9}$$

The parameter notation used was taken from [13] and [1]. When the angular parameter ψ is set to some multiple of $\pi/2$ the transformation from one frame to another is simplified. This means that the computations are easier if the robot has joints or limbs that are parallel or perpendicular to one another.

A.2 Forward kinematics

Forward kinematics is a process of using multiple transformation matrices in order to represent the articulation of a mechanism. This is done using the translation and rotation matrices described in section A.1. Combining the translation and rotation matrices gives a transformation from one link to another, described earlier by equation A.9. Therefore the transformation matrix can move a point on the link on one side of the joint to a point on the link on the other side of the joint.

Using a transformation matrix for each joint in a chain of joints the position and direction of the last link in the chain is found by simply multiplying the matrices in the order in which their corresponding joint appear in the chain of joints and links.

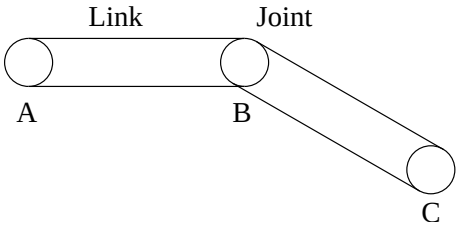

Figure A.1: Links and joints used in forward kinematics.

The solution to the forward kinematics problem is straight forward. To move two frames, $i-1$ and i, into coincidence a transformation matrix A.6 can be used. For the more general case when moving from frame 0 to frame j by homogeneous matrix operations the following applies:

$$\mathrm{T}_0^{\mathrm{j}} = \mathrm{T}_0^1 \mathrm{T}_1^2 \mathrm{T}_2^3 \ldots \mathrm{T}_{\mathrm{j}-1}^{\mathrm{j}} \tag{A.10}$$

A.3 Inverse kinematics

With inverse kinematics the position and direction of the last joint in a chain of joints is known. That is the transfer matrix function $\mathrm{T}_0^{\mathrm{n}}$ from equation A.10 is known, where n is the index referring to the last joint. Evaluating the forward kinematic matrices containing the not yet determined joint angles, leads to a four by four matrix equation, where the angles can be calculated based on the goal position. The analytical solution to one such matrix equation is seen in section 3.2.1 on page 14.