Generation of semi-optimal gait trajectories for a biped robot

Generation of semi-optimal gait trajectories for a biped robot

E.A.C.S. Peeters

DCT 2006.124

Traineeship report

Coach(es): Yuichi Tazaki

Supervisor: Prof. Jun-ichi Imura and Prof. Henk Nijmeijer

Technische Universiteit Eindhoven
Department Mechanical Engineering
Dynamics and Control Group

Eindhoven, November, 2006

Abstract

Over the last decennia there has been a wide interest in walking robots. They are potentially more capable of moving through the human environment and going over rough terrain than wheeled vehicles. We generate some 2D gaits for an approximately 0.5 [m] tall human like entertainment robot and we apply these in a very first experiment. The locomotion types we study are: walking at constant velocity, accelerating and decelerating. Unlike many others, we explicitly consider minimal energy consumption. Therefore, the approach to generate gait trajectories is the central part in this work. This work basically continues the work of Tazaki et al. [1].

Keywords: bipedal walking robots, hybrid systems, optimal control.

Contents

Chapter 1

Introduction

Over the last decennia there has been a wide interest in walking robots. They are potentially more capable of moving through the human environment and going over rough terrain than wheeled vehicles. To move through such environments, bipedal robots need to use various motions. Such motions include walking at constant velocity, accelerating, decelerating, changing walking direction and so on.

Ways to generate trajectories for robots have been studied by many authors, but only some of them have attempted to include a kind of optimality, like Tazaki et al. [1], Denk at al. [3] and Hardt et al. [4]. Tazaki et al. [1] and Denk at al. [3] concentrate on walking at different velocities and precalculate optimal partial trajectories that can be concatenated into a walk from one position and posture to another. They minimize a measure of the energy consumption of the joint actuators during the optimization of the partial trajectories as well as during the concatenation. But their methods to minimize the energy consumption are different in many ways. In contrast to Denk et al. [3] for example, Tazaki et al. [1] use an explicit solution as a building block, based on energy minimization of an approximated system.

Tazaki et al. [1] do numerical simulations. However, it is also desirable to show that the method can effectively be applied to an experimental setup. Therefore, this research focusses on applying the method to a speecys SPC001 bipedal robot, which is available on the market (see Figure 1.1). To this end, a multi-body model is developed and the numerical experiment is repeated for the Speecys robot. The calculated trajectories are applied to the robot in a first experiment. Additionally, some small changes are made to the method in order to improve the possibilities for step planning.

The report is organized a follows. First, we explain the techniques we use to derive gait trajectories in the methodology chapter. This is the central part of this report. It begins with the experimental setup, it then presents the control method and ends with the implementation of the control method. Next, we present the experimental results. Last, we briefly summarize our main findings and present some directions for further research in the conclusions and recommendations.

Figure 1.1: Speecys SPC001 robot.

Chapter 2

Methodology

The purpose of this chapter is to explain our work in such a way that the reader can understand and repeat it. Let us begin with the general idea of the previously developed method by Tazaki et al. [1] and the additions that we make. In the subsequent paragraphs the details will become clear.

To begin, we need a mathematical representation of the robotic system. Moreover, we need to define a cost function since we want to calculate energy efficient trajectories. It is only useful to use a representation and cost function for which the optimal trajectories can be calculated in an efficient way. Also, the representation must capture the dominant dynamics of the robot. First, we model the robot dynamics by a nonlinear Lagrange model. Such a model can accurately capture the dynamics of the robot. To take into account optimality, we design a cost function that represents the control input energy. This cost function must be minimized. It is quadratic in the input torque.

Because there is no explicit optimal solution available to the optimization problem consisting of the quadratic cost function combined with the nonlinear Lagrange model, we obtain a linear model by an input transformation (feedback linearization). This requires full actuation and no constraints. Doing so, the quadratic cost function becomes nonlinear instead of quadratic, since we define a new input. The nonlinear cost function must be transformed into a (piecewise defined) quadratic cost function in order to calculate the optimal trajectories explicitly. This is done by using a piecewise constant approximation of the mass matrix $\boldsymbol{M}$ and gravity/coriolis/centrifucal vector $\boldsymbol{H}$. Thus $\boldsymbol{M}$ and $\boldsymbol{H}$ are sampled. The resulting linear model with a piecewise defined quadratic control input energy cost function has an explicit solution for the optimal trajectories once the samples have been chosen.

To get a good piecewise constant approximation for $\boldsymbol{M}$ and $\boldsymbol{H}$, the sample points must be chosen in an appropriate way. It is not possible to use a dense grid because this would take too much calculation time. Therefore, a limited set of samples has to be obtained in a more efficient way. We optimize the sample points by minimizing the total control input energy of all inter-sample trajectories. The optimization is performed by sequential quadratic programming because no analytic solution is known. This requires off-line heavy computation.

The resulting trajectories are stored in a database which is used by a step sequence planner. The step sequence planner combines steps in such a way that the robot walks a specified distance within a specified time period, using as less input energy as possible. To this end, the step planner uses a depth-first-search algorithm to search through a search tree. The resulting trajectories are

traced online by the micro-controllers in each actuator (situated in the ankles, knees and hips). These actuators all have their own PID controller.

We use a slightly different method than Tazaki et al. [1]. They use a separate cost function for each walking velocity. Also, their cost function definition does not enhance the possible to directly specify a specific walking velocity. This makes it difficult to design a step sequence planner. This problem is solved by using a constrained optimization technique. The step planner itself is designed using the directed graphs introduced in the work of Tazaki et al. [1].

2.1 Experimental Setup

The Speecys SPC001 Robot (Figure 1.1) has 21 actuators. Every actuator consists of an electrical motor and a micro-controller (Futuba servo). The actuators, together with a camera and microphone, can be controlled centrally by a CPU that is attached to the robot. Eight rechargeable batteries supply the robot with power.

During this research, however, a PC is used to directly control the actuators and an external power source is used for power delivery. In this way, there is no need to learn about the operating system that controls the robot's CPU and there is no need to recharge batteries. A cable with a Serial and an USB end connects the serial ports of the actuators to the USB port of the PC. Driver software installed on the PC takes care of the conversion between USB and Serial (Figure 2.2).

Six out of 21 actuators (the ankle, knee and hip of each leg) are commanded to track the specified trajectories during a walking motion. Six other actuators of the leg are fixed in a certain position because they are used to perform motions in the direction perpendicular to both the walking direction and height of the robot. The arms and head are disassembled from the robot. The resulting system is about the simplest possible to perform 2D walking (Figure 2.1).

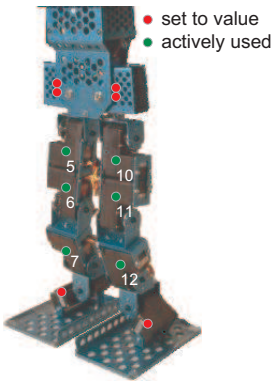

Figure 2.1: actuator activity.

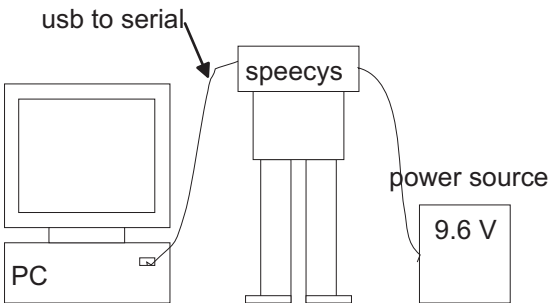

Figure 2.2: setup layout.

2.2 Nonlinear Model

To be able to calculate an optimal trajectory there is a need for a mathematical representation of the robotic system, if model based optimization is considered. In this research, first, a nonlinear multi-body model is constructed of which the parameters correspond to measurable physical parameters of the system. The robot has been disassembled in order to get the masses of all links

by using a scale. The lengths of all links have been measured by a ruler and the centers of mass are estimated. Second, an approximate model is extracted from the multi-body model for which the optimal trajectory is known. This section considers the multi-body model.

The multi-body model is derived using a Lagrange approach with relative generalized coordinates. The model consists of seven links in an open chain structure connected by 6 joints. A seventh joint connects the structure to the earth. The mass matrix is denoted by $\boldsymbol{M}$, $\boldsymbol{H}$ denotes the vector containing the centrifugal / Coriolis / gravity terms and the vector $\boldsymbol{T}$ contains the actuator torques. The column vector $\boldsymbol{q}$ contains the angles $\theta_i, i \in \{1, 2, 3, ..., 7\}$ (see Figure 2.3) and $\boldsymbol{x} = [\ \boldsymbol{q} \quad \dot{\boldsymbol{q}}\ ]^T$.

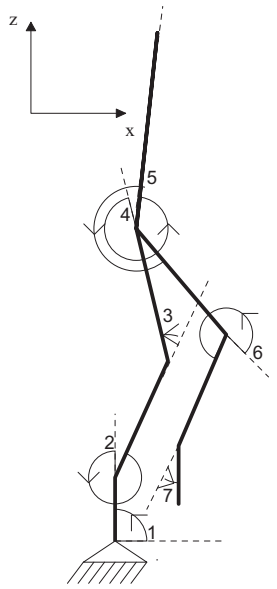

Figure 2.3: speecys SPC001 Robot conventions.

The general equations of motion are:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{T} \tag{2.1}$$

By using $\boldsymbol{F} = \boldsymbol{M}^{-1}$ and $\boldsymbol{G} = -\boldsymbol{M}^{-1}\boldsymbol{H}$ the equations of motion can also be written as:

$$\ddot{\boldsymbol{q}} = \boldsymbol{F(q)T} + \boldsymbol{G}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{2.2}$$

During walking the dynamics of the robot changes, depending on which foot has contact with the ground. A distinction can be made between left leg stance, right leg stance and both feet ground contact. The both feet ground contact mode can be further split up into left to right stance leg change and right to left stance leg change. Doing so four modes I result defined as $\boldsymbol{L}$, $\boldsymbol{R}$, $\boldsymbol{LR}$ and $\boldsymbol{RL}$ respectively. To denote the different modes the index I is added to the equations of motion.

$$\ddot{\boldsymbol{q}} = \boldsymbol{F}_I\boldsymbol{(q)T} + \boldsymbol{G}_I(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{2.3}$$

$$I \in \{\boldsymbol{L},\boldsymbol{R},\boldsymbol{LR},\boldsymbol{RL}\} \tag{2.4}$$

The motions about the left and right leg can be assumed equal if both legs are the same, only the numbering of the joints changes because the base flips from one end of the chain to the other when a next step is made.

There are several possible trajectories between an initial and final position and posture, however the trajectory with least input energy is preferable. To obtain such a trajectory the following cost function can be minimized:

$$J_{p1}(\boldsymbol{T}; h) = \int_0^h \boldsymbol{T}(\tau)^T \boldsymbol{R}_I \boldsymbol{T}(\tau) d\tau \tag{2.5}$$

The choice of $\mathbf{R}_I$ determines which inputs are minimized most during the optimization. We choose a unity matrix, so that there is no difference between the input weights. The total nonlinear problem can now be stated as follows:

Problem 1 *Given initial state $\boldsymbol{x}_0$, final state $\boldsymbol{x}_f$, modes $I \in \{\boldsymbol{L}, \boldsymbol{R}, \boldsymbol{LR}, \boldsymbol{RL}\}$ and weight $\boldsymbol{R}_I > 0$ find an input $\boldsymbol{T}$ and transition time $h > 0$ that minimize the cost function:*

$$J_{p1}(\boldsymbol{T}; h) = \int_0^h \boldsymbol{T}(\tau)^T \boldsymbol{R}_I \boldsymbol{T}(\tau) d\tau \tag{2.6}$$

subject to:

$$\ddot{\boldsymbol{q}} = \boldsymbol{F}_I(\boldsymbol{q})\boldsymbol{T} + \boldsymbol{G}_I(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{2.7}$$

In general this nonlinear problem cannot be solved explicitly, therefore the problem is transformed into an approximate problem.

2.3 Approximate model

The optimal control input for the full nonlinear model is not known. Therefore, an approximate model with a known optimal control is extracted from the nonlinear model. The approximate model structure is a linear system with a piecewise defined quadratic input energy cost function. To obtain a linear system dynamics an input transformation is performed (also referred to as feedback linearization):

$$\boldsymbol{T} = \boldsymbol{F}_I(\boldsymbol{q})^{-1}(\boldsymbol{v} - \boldsymbol{G}_I(\boldsymbol{q}, \dot{\boldsymbol{q}})) \tag{2.8}$$

The input transformation requires $\boldsymbol{F}_I$ to be regular. The resulting system dynamics are:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{v} \qquad \boldsymbol{A} = \begin{pmatrix} \boldsymbol{O} & \boldsymbol{I} \\ \boldsymbol{O} & \boldsymbol{O} \end{pmatrix} \qquad \boldsymbol{B} = \begin{pmatrix} \boldsymbol{O} \\ \boldsymbol{I} \end{pmatrix} \tag{2.9}$$

The dimension of $\boldsymbol{x}$ is 14×1, the dimension of $\boldsymbol{A}$ is 14×14, the dimension of $\boldsymbol{B}$ is 14×7 and the dimension of $\boldsymbol{v}$ is 7×1. Note that the obtained linear dynamics is the same for each mode. The input transformation, however, can be different for each mode.

Next, a piecewise defined quadratic cost function is required to be able to get an explicit solution for the optimal trajectory. To this end $\boldsymbol{F}(\boldsymbol{q}(\tau))$ and $\boldsymbol{G}(\boldsymbol{q}(\tau), \dot{\boldsymbol{q}}(\tau))$ are evaluated at a finite set of time instants t_k, $k \in \{1, 2, 3, ..., N_s\}$ separated by time intervals h_k, $k \in \{1, 2, 3, ..., N_s - 1\}$.

$$\boldsymbol{q}(t_k) = \frac{\boldsymbol{q}_k + \boldsymbol{q}_{k+1}}{2} \qquad \dot{\boldsymbol{q}}(t_k) = \frac{\dot{\boldsymbol{q}}_k + \dot{\mathbf{q}}_{k+1}}{2} \tag{2.10}$$

Doing so, constant $\boldsymbol{F}$ and $\boldsymbol{G}$ result, $\widehat{\boldsymbol{F}}$ and $\widehat{\boldsymbol{G}}$ respectively.

To further simplify the problem it is imposed that no mode change may occur between $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$, $k \in \{1, 2, 3, ..., N_s - 1\}$. Thus at every mode change a sample ($\boldsymbol{x}$) must be placed. This condition is called the mode invariance condition. A new approximate problem results (2):

Problem 2 *Given initial state $\boldsymbol{x}_1$, intermediate states $\boldsymbol{x}_k, k \in \{2,3,...,N_s-1\}$, final state $\boldsymbol{x}_{N_s}$, modes $I \in \{\boldsymbol{L},\boldsymbol{R},\boldsymbol{LR},\boldsymbol{RL}\}$ and weights $\boldsymbol{R}_I > 0$, find an input $\boldsymbol{T}$ and transition times $h_k > 0, k \in \{1,2,3,...,N_s-1\}$ that minimize the cost function:*

$$J_{p2}(\boldsymbol{v};h_k) = \int_0^{h_k} (\boldsymbol{v}(\tau) - \widehat{\boldsymbol{G}}_I(\boldsymbol{q}_k,\boldsymbol{q}_{k+1}))^T \widehat{\boldsymbol{F}}_I(\boldsymbol{q}_k,\boldsymbol{q}_{k+1})^{-T} \boldsymbol{R}_I \widehat{\boldsymbol{F}}_I(\boldsymbol{q}_k,\boldsymbol{q}_{k+1})^{-1} (\boldsymbol{v}(\tau) - \widehat{\boldsymbol{G}}_I(\boldsymbol{q}_k,\boldsymbol{q}_{k+1}))d\tau \tag{2.11}$$

subject to:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{v} \qquad \boldsymbol{A} = \begin{pmatrix} \boldsymbol{O} & \boldsymbol{I} \\ \boldsymbol{O} & \boldsymbol{O} \end{pmatrix} \qquad \boldsymbol{B} = \begin{pmatrix} \boldsymbol{O} \\ \boldsymbol{I} \end{pmatrix} \tag{2.12}$$

The solution of the approximate problem between $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$ $k \in \{1,2,3,...,N_s-1\}$ is (taken from [1],* denotes optimality):

Theorem 1 (solution problem 2)

$$\boldsymbol{x}^* = \begin{pmatrix} \boldsymbol{q}^* \\ \dot{\boldsymbol{q}}^* \end{pmatrix} \tag{2.13}$$

$$\boldsymbol{v}^* = \ddot{\boldsymbol{q}}^* \tag{2.14}$$

$$\boldsymbol{q}^* = \begin{pmatrix} \boldsymbol{q}_k & \dot{\boldsymbol{q}}_k & \boldsymbol{q}_{k+1} & \dot{\boldsymbol{q}}_{k+1} \end{pmatrix} \begin{pmatrix} 1 & 0 & -\frac{3}{h_k^2} & \frac{2}{h_k^3} \\ 0 & 1 & -\frac{2}{h_k} & \frac{1}{h_k^2} \\ 0 & 0 & \frac{3}{h_k^2} & -\frac{2}{h_k^3} \\ 0 & 0 & -\frac{1}{h_k} & \frac{1}{h_k^2} \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix} \tag{2.15}$$

$$J_{p2}^* = \begin{pmatrix} \boldsymbol{q}_k & \dot{\boldsymbol{q}}_k & \boldsymbol{q}_{k+1} & \dot{\boldsymbol{q}}_{k+1} & \widehat{\boldsymbol{G}}_I \end{pmatrix} \begin{pmatrix} \widehat{\boldsymbol{R}}_I\frac{12}{h_k^3} & \widehat{\boldsymbol{R}}_I\frac{6}{h_k^2} & -\widehat{\boldsymbol{R}}_I\frac{12}{h_k^3} & \widehat{\boldsymbol{R}}_I\frac{6}{h_k^2} & 0 \\ \widehat{\boldsymbol{R}}_I\frac{6}{h_k^2} & \widehat{\boldsymbol{R}}_I\frac{4}{h_k} & -\widehat{\boldsymbol{R}}_I\frac{6}{h_k^2} & \widehat{\boldsymbol{R}}_I\frac{2}{h_k} & \widehat{\boldsymbol{R}}_I \\ -\widehat{\boldsymbol{R}}_I\frac{12}{h_k^3} & -\widehat{\boldsymbol{R}}_I\frac{6}{h_k^2} & \widehat{\boldsymbol{R}}_I\frac{12}{h_k^3} & -\widehat{\boldsymbol{R}}_I\frac{6}{h_k^2} & 0 \\ \widehat{\boldsymbol{R}}_I\frac{6}{h_k^2} & \widehat{\boldsymbol{R}}_I\frac{2}{h_k} & -\widehat{\boldsymbol{R}}_I\frac{6}{h_k^2} & \widehat{\boldsymbol{R}}_I\frac{4}{h_k} & -\widehat{\boldsymbol{R}}_I \\ 0 & \widehat{\boldsymbol{R}}_I & 0 & -\widehat{\boldsymbol{R}}_I & \widehat{\boldsymbol{R}}_I h_k \end{pmatrix} \begin{pmatrix} \boldsymbol{q}_k \\ \dot{\boldsymbol{q}}_k \\ \boldsymbol{q}_{k+1} \\ \dot{\boldsymbol{q}}_{k+1} \\ \widehat{\boldsymbol{G}}_I \end{pmatrix} \tag{2.16}$$

$$\widehat{\boldsymbol{R}}_I \equiv \widehat{\boldsymbol{F}}_I^{-T} \boldsymbol{R} \widehat{\boldsymbol{F}}_I^{-1} \tag{2.17}$$

This solution requires that $\boldsymbol{x}_k$, $k \in \{1,2,3,...,N_s\}$ and h_k, $k \in \{1,2,3,...,N_s-1\}$ are given. Note that the solution for the optimal acceleration of the approximate problem $\boldsymbol{v}^* = \ddot{\boldsymbol{q}}^*$ does not depend on the weight of the cost function. It is an approximate solution because of the piecewise constant approximation of $\boldsymbol{M}$ and $\boldsymbol{H}$. The required torque to approximately realize this acceleration can be computed. In general the sets h_k and $\boldsymbol{x}_k$ are not known. Next section considers how to choose these parameters.

2.4 Optimization constant velocity walking

In chapter 2.3 the semi-optimal trajectory between the samples $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$, $k \in \{1,2,3,...,N_S-1\}$ has been derived. At this point freedom remains to choose $\boldsymbol{x}_k$, $k \in \{1,2,3,...,N_s\}$ and h_k, $k \in \{1,2,3,...,N_s-1\}$. We choose the samples and transition times such that the energy input of

the approximated problem is minimal. To this end, the following cost function which represents the sum of all costs between samples can be minimized (see also Figure 2.4):

$$J_{sum} = \sum_{k=1}^{N_s-1} J_{p2}^*(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k) \tag{2.18}$$

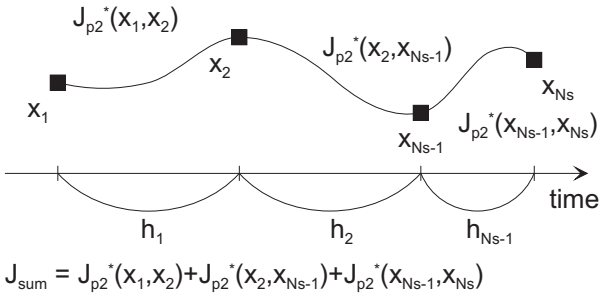

Figure 2.4: constant velocity optimization problem.

The number of samples N_s still needs to be chosen. Using more samples brings the approximation closer to the nonlinear model and results in a smaller J_{sum}. However, the calculation time increases enormously when the number of variables increases. One can try to increase the number of variables until no significant decrease in J_{sum} can be noticed.

We take a minimum number of samples in order to have a minimal calculation time. There must be a sample at each mode change. During one walking cycle four modes $\boldsymbol{R} \rightarrow \boldsymbol{RL} \rightarrow \boldsymbol{L} \rightarrow \boldsymbol{LR}$ are passed. During steady state walking the only difference between to two subsequences $\boldsymbol{R} \rightarrow \boldsymbol{RL}$ and $\boldsymbol{L} \rightarrow \boldsymbol{LR}$ is that a different leg is swinging / standing. Thus, only the numbering of the actuators needs to be changed. Therefore, in the optimization only one of these two subsequences needs to be considered. Doing so, two modes are left and because the modes are connected into a cycle there are only two different boundaries at which a sample must be placed. To prevent ground scuffing of the swing foot an additional sample point is placed in the swing mode, so $N_s = 3$.

In addition to the cost function and accompanying equations of motion, some constraints must be considered to prevent collisions and falling during walking. Also, constraints can be used to guarantee some desired properties of a motion, like walking velocity, step length, height of upper body, etcetera. Last, constraints can reduce the number of variables to be optimized.

Let us begin with the constraints in angle coordinates. The feet must always be parallel to the ground and the upper-body must always make an inclination angle α to the vertical, more precisely:

- $\theta_1 = \pi/2$
- $\theta_4 = -\theta_2 - \theta_3 - \alpha$
- $\theta_7 = \pi + \alpha - \theta_5 - \theta_6$

This set of constraints reduces the number of variables to be optimized.

Some constraints have a simpler formulation when they are formulated in an alternative coordinate set. Therefore, a coordinate transformation is applied. This coordinate transformation transforms the angle coordinates into position coordinates. After the transformation a reduced set of variables $\boldsymbol{P}^k = [\ \boldsymbol{p}^k \quad \dot{\boldsymbol{p}}^k\ ]^T = [\ {p_{x1}}^k \quad {p_{z1}}^k \quad {p_{x2}}^k \quad {p_{z2}}^k \quad \dot{p}^k_{x1} \quad \dot{p}^k_{z1} \quad \dot{p}^k_{x2} \quad \dot{p}^k_{z2}\ ]^T$ (Figure 2.5) is left, where $k \in \{1, 2, 3, ..., N_s\}$ denotes the sample number.

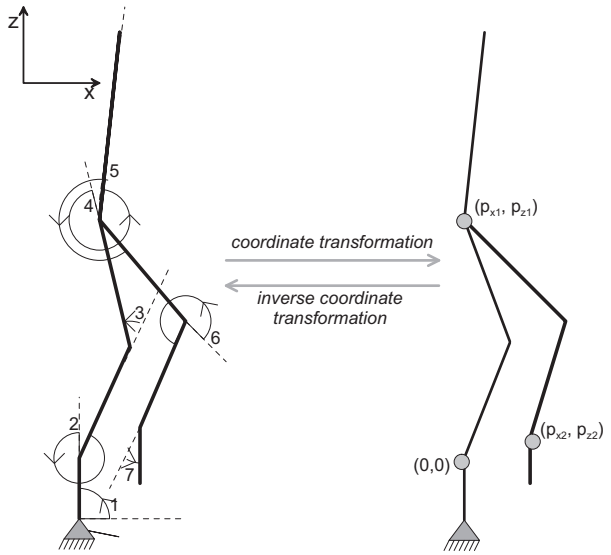

Figure 2.5: coordinate transformation.

The inverse of this transformation transform position coordinates into angle coordinates. We need to calculate this inverse many times during the optimization. We calculate it as follows. The angles of the knees are determined by using the distances between the hip and the two ankles. When the length of the leg parts are included, some triangles can be formed with known side lengths. There are two possible triangles, but only one is feasible, because the knee is only allowed to kink in one direction, like the human knee. The lengths of all sides in this triangle are known, accordingly all angles can be calculated. A difficulty, however, is that not all combinations of positions are allowed because of limited link lengths. In the implemented inverse transformation the angles get imaginary parts when the lengths are infeasible. This imaginary part of the angles is constrained to be zero during the optimization, so that no infeasible solutions are possible. In addition to the position transformation, a transformation is necessary that transforms the angular velocities into translational velocities of the new coordinates.

When three samples are used the set of variables to optimize becomes:
$S = [\ \boldsymbol{P}^{1T} \quad \boldsymbol{P}^{2T} \quad \boldsymbol{P}^{3T} \quad h_1 \quad h_2\ ]^T$. These are already 26 variables. However, as can be seen in Figure 2.6, the postures at sample 1 and 3 are the same, only the legs are interchanged. This property can be used to decrease the number of variables by imposing the following constraints:

- ${p_{x1}}^3 = -({p_{x2}}^1 - {p_{x1}}^1)$
- ${p_{z1}}^3 = {p_{z1}}^1$
- ${p_{x2}}^3 = -{p_{x2}}^1$
- ${p_{z2}}^3 = {p_{z2}}^1 = 0$

So 21 variables are left $S = [\ {p_{x1}}^1 \quad {p_{z1}}^1 \quad {p_{x2}}^1 \quad \dot{\mathbf{p}}^{1^T} \quad \boldsymbol{P}^{2^T} \quad \dot{\boldsymbol{p}}^{3^T} \quad h_1 \quad h_2\ ]^T$. To prevent ground scuffing, the swing leg must be above a certain height at the intermediate sample: $p_{z2}^2 > 0$. Performing the optimization without imposing an upper and lower bound on the other variables reveals that there are many local optima, to which one can go by changing the initial guess of the SQP algorithm. However, most of these optima cannot be implemented, because the obtained gaits lead to foot collisions or unrealistically high velocities and accelerations. Therefore, bounds ensure that the algorithm converges to an optimum that can be implemented to the robot. These bounds also dramatically decrease computation time.

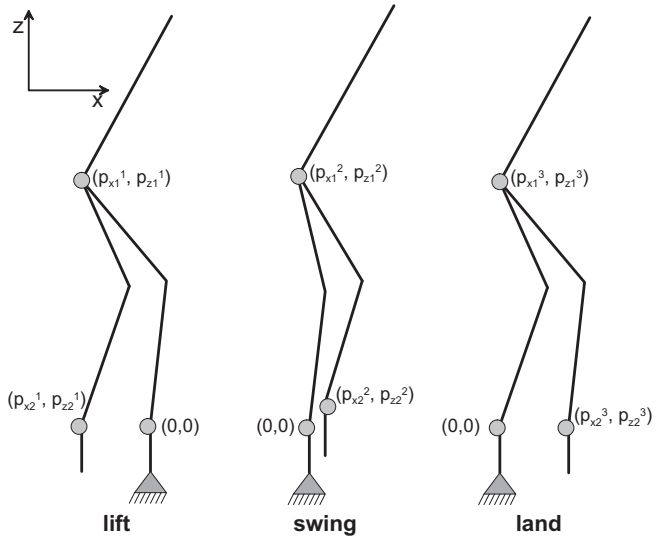

Figure 2.6: constant velocity optimization settings.

Until now the constraints are only applied to the samples. During the trajectories between the samples these constraints can still be violated. There is an explicit solution for these intermediate trajectories (theorem 1). This information can in principle be used to also apply the constraints to the intermediate trajectories. However, these expressions are very involved. Applying "hard" constraints then leads to difficulties finding appropriate initial guesses for the optimization algorithm. To circumvent this difficulty, penalties are added to the cost function. Even if the cost function is initially penalized, the solution will often converge along the slopes of the penalty function to a solution with a non-penalized cost function. The penalties are evaluated at five equally spaced points between all samples, starting at $h_k/4$ and ending at $3h_k/4$. They include:

- the swing leg must be above the ground
- the swing leg must be below a certain height
- the upper body must be above a certain height

If only these constraints are used, often a solution results in which the swing leg moved backwards. This seems quite unnatural. Therefore, $\dot{\theta}_5$, the angular velocity of the angle between the upper-body and the swing leg, is constrained to be positive during the intermediate trajectories. It is, however, possible to land or takeoff the swing foot in a backward movement because the beginning and end of the intermediate trajectory is not penalized. In future research it is

highly desirable to take dynamically stable walking into consideration, e.g. by constraints on the zero moment point. In this way, some unnatural constraints, that are necessary now to obtain a feasible solution, can be removed and stable walking is always guaranteed.

2.5 Step sequence optimization

Suppose the robot is equipped with a vision camera and sees an interesting object ten meters ahead. It wants to go there to have a better look, but it also has to take care about its limited time and energy resources. The step sequence and walking velocity must be chosen such that it ends near the object in an efficient way. The step sequence optimization routine takes care of this task. It uses knowledge about possible steps and combines these steps such that the objective is met using as less energy as possible.

In chapter 2.4 constant velocity walking is derived. The new task requires that the robot is also to be able to start and stop. Moreover, it must be able to switch from one walking velocity to another. To this end, the optimization as presented in chapter 2.4 is performed for a set of walking velocities. A set of optimized sample points belongs to each walking velocity. The sample points of different walking velocities can be connected by optimized inter-sample trajectories which can also be calculated using theorem 2. However, care should be taken that no mode change occurs during these switching motions and that no constraints are violated. Also, a reasonable choice must be made for the transition time. The robot can switch from one speed to another along these trajectories. The set of allowable trajectories can be visualized in a directed graph as can be seen in Figure 2.7 (by [1]). In this graph the cyclic nature of motions can be observed. $v_s p_k$ denotes sample number k out of an optimized sample set belonging to constant walking speed number s. We select three velocities: standstill, 0.02 $[ms^{-1}]$ and 0.04 $[ms^{-1}]$. To each velocity, except standstill, belongs a set of three samples.

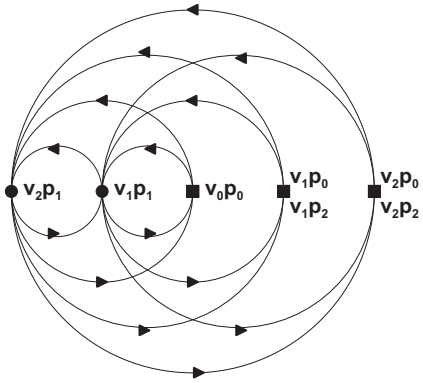

Figure 2.7: directed graph.

Each inter-sample trajectory has a known cost, distance and duration. A depth-first search algorithm evaluates each possible sample sequence to go to the interesting object. A sequence is combined until the final distance is met. A sequence is only stored if the elapsed time is less than the specified time, the final posture equals the desired final posture and the cost is less than the previously stored cost. The algorithm evaluates all possible sequences that make up the specified distance in order to obtain the global optimal sample sequence. In Figures 2.8 and 2.9 the solution of an example problem is visualized by a search tree and an alternative directed graph.

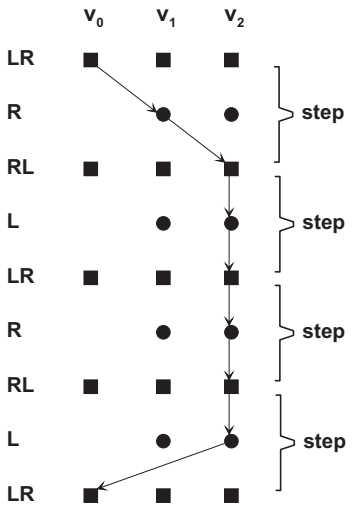

Figure 2.8: alternative directed graph.

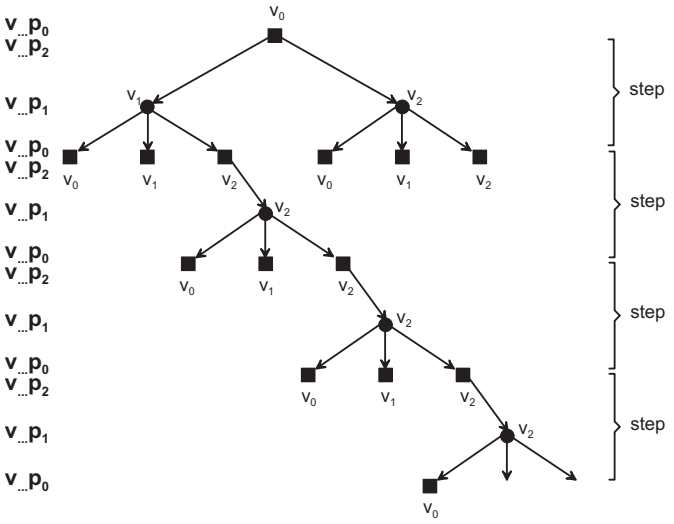

Figure 2.9: search tree.

2.6 Implementation

The implementation of the proposed method is the most time consuming part because many details need to be considered. This report sticks to the main issues and some important details. The main challenge is to keep the calculation time within reasonable limits. There are essentially three tasks that must be done. First, the optimization for constant velocity walking needs to be performed to obtain the optimal constant velocity samples. This set of samples is used in the step sequence optimization. Second, the step sequence optimization must be performed to obtain the optimal step sequence. Third, the obtained step sequence must be translated into angle trajectories that can be traced by the robot actuators. Last, the actuator angles must be measured while the robot is tracing the desired trajectories and these measured angles must be compared to the desired ones. Figure 2.10 displays the general layout of the implementation.

The implementation is performed in Matlab and C++. Matlab has the advantage that the coding is relatively easy because many building blocks are available in the Matlab library. However, the disadvantage is that running the code takes much computation time, since the code needs to be interpreted while running. In C++, the code is compiled before running so that the code runs more efficiently. The constant velocity sample optimization is performed in Matlab by constrained sequential quadratic programming (SQP). This algorithm is often used for general nonlinear constrained problems and is readily available in the Matlab optimization toolbox. The cost function and the constraints must be supplied to this algorithm. The cost function contains $\boldsymbol{M}$ and $\boldsymbol{H}$. $\boldsymbol{M}$ and $\boldsymbol{H}$ are calculated symbolically, this takes about 3 minutes on a 1400MHz cpu. A symbolic expression has the advantage that $\boldsymbol{M}$ and $\boldsymbol{H}$ only have to be calculated once and can then be evaluated. In this way they do not need to be completely recalculated numerically during each iteration. However, the expressions are very involved and even evaluation takes a lot of computation time. To reduce this computation time, $\boldsymbol{M}$ and $\boldsymbol{H}$ are implemented in a Matlab mex function. A mex function is written in C and is compiled before execution. The optimization of the samples set belonging to one velocity takes about 40 minutes. In this research an optimization is performed for 0.02 and 0.04 $[ms^{-1}]$.

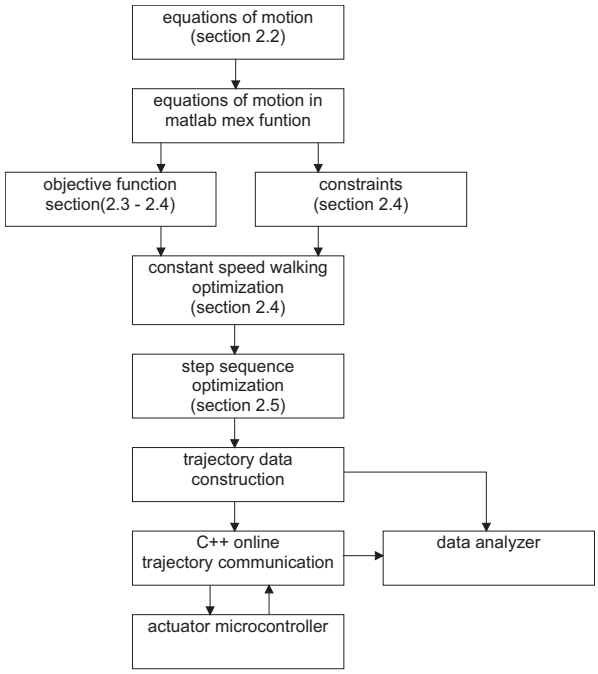

Figure 2.10: implementation diagram.

When the samples are available, evaluating the symbolic expression for the trajectories between the samples takes just a few seconds. These trajectories must be combined by a step sequence planner in order to fulfill the walking objective. The step sequence algorithm is based on depth-first-search with unknown depth. This algorithm is written in Matlab. The result of this step sequence optimization is a series of symbolic expression for the trajectories. These expressions are evaluated at a number of time instances (discretized). Then the values are translated into the appropriate values for the actuators and stored in a text file. This text file is red by a C++ programm that sends these values online to the micro controllers of the actuators. At a fixed interval, the C++ programm asks the micro-controllers to send the present angle of the actuator. This measurement result is written to a different text file, which can be red by Matlab. Eventually, the calculated trajectories and the measurements are compared in Matlab.

Chapter 3

Experimental Results

Here we present and discuss the experimental results. The objective of the experiments is: demonstrate the effectiveness of the proposed control method. For this, we want to answer two questions. (1) Is the optimization scheme able to come up with gait trajectories that let the robot walk? (2) Are the gait trajectories that we obtain truly optimal? Two experiments are carried out in order to answer these questions.

First, we want to demonstrate that the robot tracks the trajectories obtained by numerical optimization for both constant velocity walking and the switching motion. To answer question 1, we want to demonstrate that the robot actually walks. If it walks, we can conclude that the optimization routine is able to come up with gait trajectories that let the robot walk. To answer question 2, we want to demonstrate that the calculated trajectory is truly optimal by showing that the torque input for the measured motion is less than the torque input for a slightly different trajectory. If so, that demonstrates that the obtained gait is at least locally optimal.

3.1 The experiments

We use the constant velocity optimization routine to obtain gait trajectories for walking at 0.02 and 0.04 $[ms^{-1}]$. The robot actuators are commanded to track these trajectories in two separate experiments. This report discusses the result for 0.04 $[ms^{-1}]$. Next, we obtain a gait trajectory for accelerating form standstill, walking and decelerating to standstill over a distance of 0.16 $[m]$ in less than 4 [s] by using the step sequence optimization routine (see Figures 2.8 and 2.9). The step sequence optimization uses the sets of waypoints that are obtained by the two constant velocity walking optimizations. Again, the robot actuators are commanded to track the trajectories in an experiment.

Figures 3.2 and 3.3 show the results for constant velocity walking and the switching motion respectively. The dashed lines denote the calculated trajectories which the actuators must trace. The actuators send a series of measurements to the PC, the measurements are connected by lines and are shown as the solid lines in the figures. Each sub-figure shows the result of one actuator for one walking cycle, because data communication delays make it impossible to read all data within one walking cycle. One walking cycle for constant velocity walking consist of one right and one left leg step. One walking cycle for the switching motion consists of four steps (see Figure 2.8).

What do we observe in Figures 3.2 and 3.3 with respect to the tracking accuracy? We can see that the maximum tracking error is in the order of 0.1 [rad]. Also, if we look carefully, we can

see that the measured curves seem to have discontinuities, because we have connected a rather limited number of measurement data by straight lines. Further, we can see that the shape of the measurement curve is approximately the same as the shape of the simulation result curve; the measurements seem to lag behind the simulation result. This is also what we expect, because we attempt to track the simulation results. Last, the effects of initial conditions are hardly visible. That is because most measurements are initiated after a few step series.

What do we observe on the physical robot during the experiments? We here mention what stands out most when we look at the robot movements. The robot walks without support. It can accelerate, walk and stop. But, the feet never loose contact with the ground while walking. Further, the robot motions are not smooth, the robot is vibrating. Last, the actual robot movements take more time than the figures suggest. The time displayed in the figures is thus not real-time.

Can we answer questions (1) and (2) with these results? (1) The optimization method is able to generate gait trajectories that make the robot walk. But, some measures must be taken to get a more satisfactory robot gait. (2) At this moment we cannot concretely demonstrate the energy efficiency of the control method, because the tracking performance is simply not good enough to show that slightly different trajectories require more energy input.

In conclusion, the robot is able to walk, but this walk is not yet satisfactory. Also, that is not what we expect from a very first experiment. It is likely that, after a few adjustments to the robot hardware and software, the tracking performance can be significantly increased. This is necessary to get reproducible results and to demonstrate the energy efficiency of the control method.

3.2 Improving the experiments

Let us now identify the setup limitations and discuss in detail what we can do to overcome them. At this moment some hardware limitations make it difficult to obtain more accurate measurements. (1) It is only possible to send a desired position plus settling time to the micro-controllers; it is not possible to send a desired velocity. (2) The number of desired positions that can be send to the micro-controllers per second is severely limited by the low data communication rate. (3) The robot feet do not loose contact with the ground when 2D gait trajectories are applied.

As a result of limitation (1), the micro-controller forces the actuator to stop at each position we specify. This undesirable result is partly overcome by sending a new desired position to the micro-controller before the previous desired position is reached. To be more precise, the new position is sent to the actuator after the desired settling time minus the deceleration time. This deceleration time can be set for each actuator and can thus be used. This measure, however, introduces an error, because it distorts the original desired trajectory. Therefore, it is better to use a micro-controller that allows direct control over velocity or acceleration.

Due to limitation (2), the trajectories that are stored in the micro-controllers are different from the desired trajectories. Also, the number of measurements we can do per second is limited. We, therefore, only measure the angles of one actuator each step. Thus, the same step must be performed six times to obtain the data of all six actuators. If more data per step are required in order to reduce the interpolation error, multiple sequences of six steps are necessary. Although it is possible to do measurements in this way, it is not possible to sent more data to the micro-controllers per step in order to specify the desired trajectories more accurately. Therefore, it is necessary to install a faster communication system .

Limitation (3) makes that, if one foot is lifted, some tilting of the ankle and upper-body in the third dimension is necessary to keep the robot upright. However, 2D walking does by definition not include tilting in the third dimension. Thus, a 3D optimization must be performed or other measures must be taken. We make the robot walk in small steps. In this case both feet remain in contact with the ground and the robot shuffles forward. Another possibility is to modify the shape of the shoes, such that the center of gravity always remains above the shoes (see Figure 3.1)(Ravi Gondhalekar).

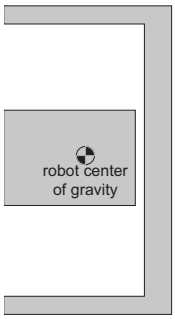

Figure 3.1: this robot shoe design keeps the COM above the shoes during 2D walking

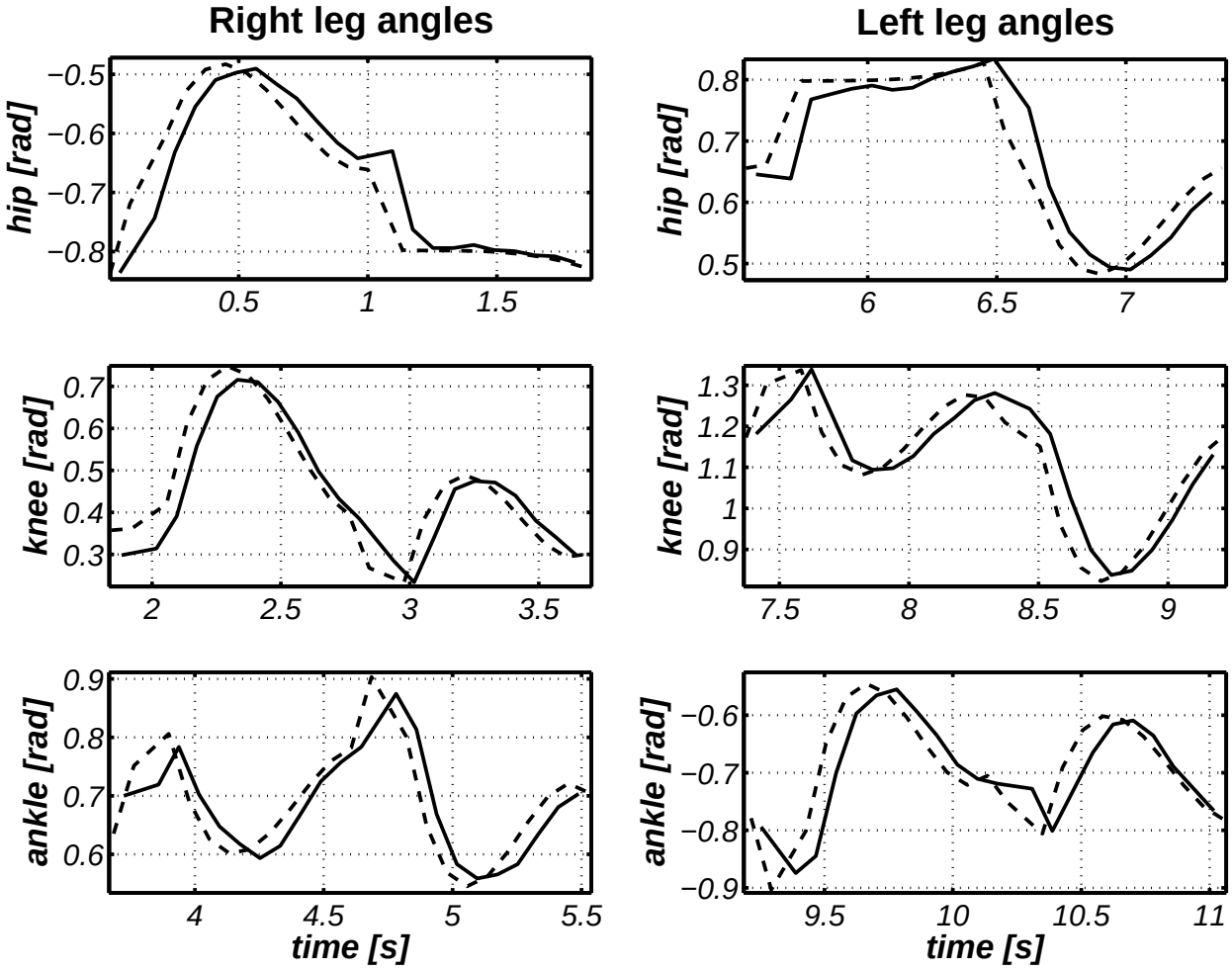

Figure 3.2: constant velocity walking 0.04 $[ms^{-1}]$. The dashed line denotes the calculated trajectory which the actuators must trace; and the solid line denotes the measured trajectory. In each plot the robot makes one left and one right leg step respectively. The data of only one actuator is measured simultaneously.

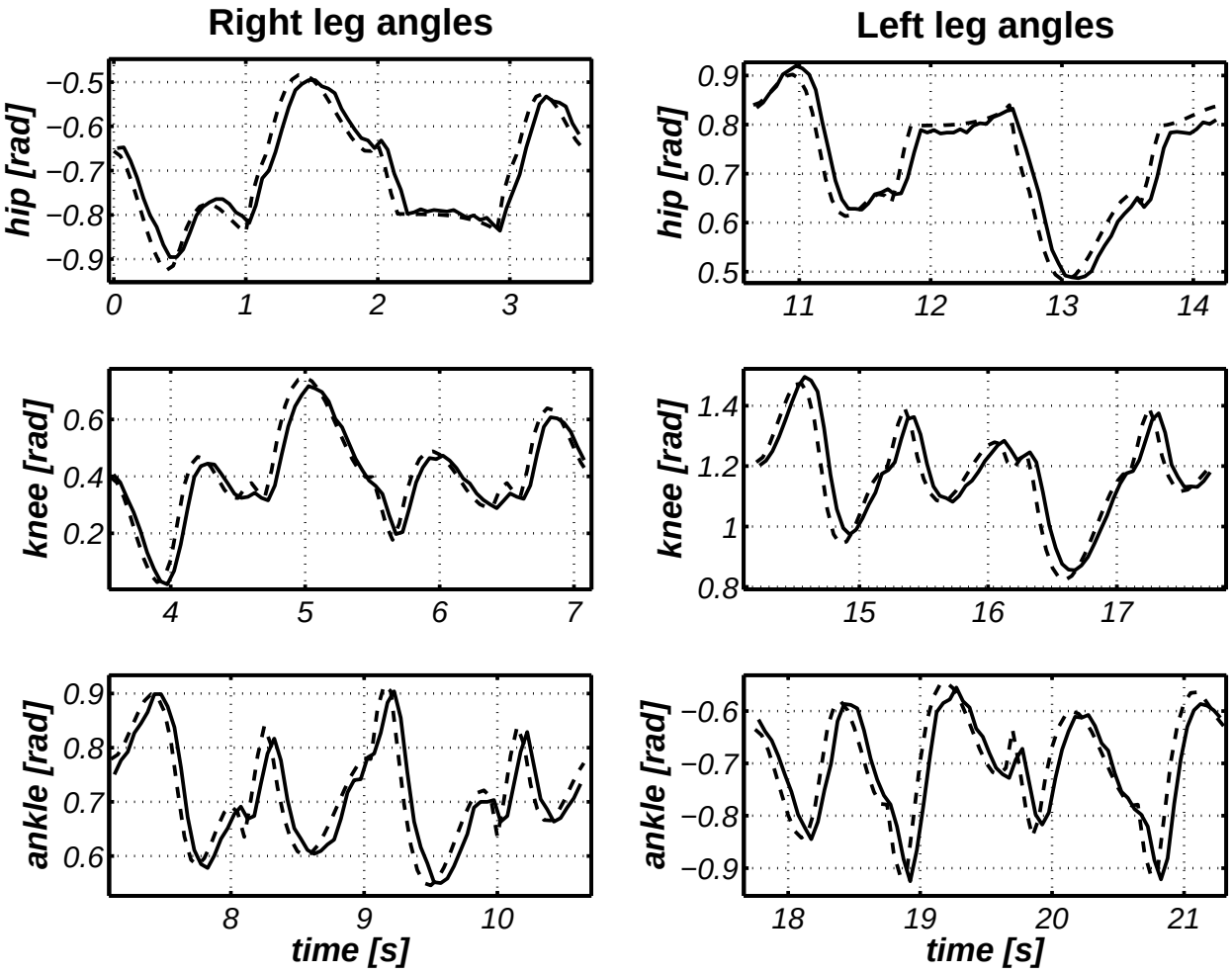

Figure 3.3: start, walk and stop. The robot is commanded to start, walk and stop over a distance of 0.16 [m] in less than 4 [s]. Each plot displays such a step sequence that consists of four steps (see Figure 2.8). The dashed line denotes the calculated trajectory which the actuators must trace; and the solid line denotes the measured trajectory. The data of only one actuator is measured simultaneously.

Chapter 4

Conclusions and Recommendations

4.1 Conclusions

This research has focussed on calculating semi-optimal 2D walking trajectories for a speecys SPC001-robot and applying them to the robot. Trajectories have been calculated for two walking velocities and for switching movements among these velocities. These trajectories have been implemented on the robot in a very first experiment. Two additions have been made to the method of Tazaki et al. [1]. (1) The possibility to directly specify velocities has been added. (2) The step sequence optimization has been adjusted such that the desired walking distance and time can be directly specified.

The robot can walk, tracing the calculated optimal trajectories for both constant velocity and steady state walking with an error of about 0.1 [rad]. Because of this error it is difficult to draw well supported conclusions about the effectiveness of the proposed new control method. The current experimental setup has some limitations that need to be taken away in order to improve the experiments. The two most important limitations are: (1) the micro-controller structure does not allow to specify velocities and (2) the maximal communication rate between the actuators and the PC is too low. These limitations are thus clearly identified and can be resolved. If these limitations are resolved it is likely that the actuators will trace the trajectories in a reasonable way. If in addition the shape of the shoes is adapted such that 2D stable walking is possible, than it can be investigated if the calculated optimal trajectories are indeed the optimal trajectories.

4.2 Recommendations

The calculated optimal trajectories are likely to deviate from the real optimal trajectories because (1) parameter inaccuracies, (2) not all physical effects are included and (3) approximations are made during the optimization. However, the objective is to minimize the energy input of the real system. Therefore, it is most natural to use data from direct online measurements in order to improve the trajectories online. As a parametrization of the trajectories the same setting as for the optimization can be used. Thus a set of samples together with known optimal trajectories between the samples parameterize the problem. The advantage of this parametrization is that it is based on some kind of optimality.

The set of samples can be re-optimized such that the measured control input energy becomes minimal. Also, additional sample points can be added. The modelbased optimized set can be

used as a starting set. During the life of the robot continuing re-optimization will then improve the robot's walking capabilities by improving the motion set. This updated motion set can be used by the step planning algorithm. To assure that only feasible motions result and to assure that there is always a feasible step sequence for any reasonable walking goal, some constraints must be considered during online re-optimization. One possibility is to fix the samples that represent the two feet ground contact modes (see Figure 4.1). For example, by a sequence ordered by step length: 0.005, 0.01, 0.02, 0.04, and 0.08 [m]. In this example each distance is covered with a precision of 0.0025 [m].

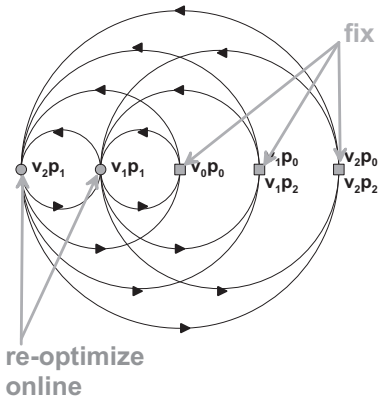

Figure 4.1: online non-model based re-optimization.

Acknowledgements

I am very grateful to Yuichi Tazaki for his valuable assistance. I like to thank Professor Jun-ichi Imura for welcoming me in his laboratory and professor Henk Nijmeijer for giving me the opportunity to do this internship. I like to thank all laboratory members for their warm welcome. I had a very pleasant time at Tokyo Institute of Technology. Further, I gratefully received the accommodation from TOKODAI, the WTB funding from the Mechanical Engineering Department and the TU/e funding from Eindhoven University of Technology.

Bibliography

[1] Yuichi Tazaki, Jun-ichi Imura (2006). Graph-based Model Predictive Control of a Planar Bipedal Robot. 17th int. Symposium on Mathematical Theory of Networks and Systems, Kyoto, Japan, pp. 128-133.

[2] Jun-ichi Imura (2004). Optimal Control of Sampled-data Piecewise Affine Systems. Automatica, Vol.40, No.40, pp.661-669.

[3] J. Denk, G. Schmidt (2003). Synthesis of Walking Primitive Databases for Biped Robots in 3D-Environments. Proc. IEEE int. conference on Robotics and Control.

[4] Michael Hardt, Oskar von Stryk, Dirk Wollherr and Martin Buss (2003). Development and Control of Autonomous, Biped Locomotion using Efficient Modeling, Simulation, and Optimization Techniques. Proc. IEEE int. conference on Robotics and Control, Taipei, Taiwan, pp. 1356-1361.

[5] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kazuhito Yokoi and Hirohisa Hirukawa (2002). A Realtime Pattern Generator for Biped Walking. Proc. 2002 IEEE int. conference on Robotics and Automation, Washington D.C., U.S.

[6] Roy Featherstone, David Orin (2002). Robot Dynamics: Equations and Algorithms. Proc. IEEE int. conference Robotics and Automation, San Fransisco, pp. 826-834.

Symbols

Symbol	Quantity	Unit
α	angle	$[rad]$
h	time	$[s]$
$\boldsymbol{H}$	gravity / centrifugal / coriolis vector	$[Nm]$
I	modes	[-]
$\boldsymbol{I}$	unit matrix	[-]
J_{p1}	cost problem 1	$[N^2m^2s]$
J_{p2}	cost problem 2	$[N^2m^2s]$
J_{sum}	summed cost	$[N^2m^2s]$
$\boldsymbol{M}$	mass matrix	$[kgm^2]$
N_s	number of samples	[-]
$\boldsymbol{p}$	reduced position vector	$[m]$
$\boldsymbol{P}$	reduced position and velocity vector	$[m]$ and $[ms^{-1}]$
$\boldsymbol{q}$	angle vector	$[rad]$
$\boldsymbol{R}$	weight matrix	[-]
t	time	$[s]$
$\boldsymbol{T}$	input torque vector	$[Nm]$
θ	angle	$[rad]$
$\boldsymbol{v}$	input vector feedback linearized system	$[Nkg^{-1}m^{-1}]$
$\boldsymbol{x}$	vector containing angles and angular velocities	$[rad]$ and $[rads^{-1}]$