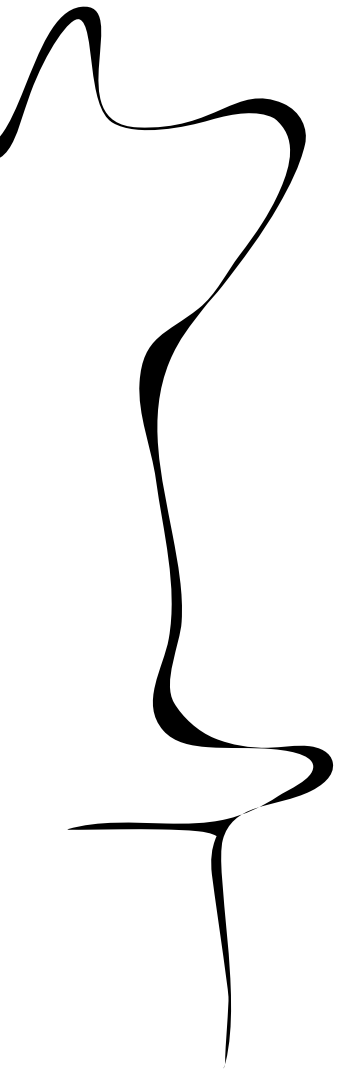
Faculty of Engineering Technology
Department of Biomechanical Engineering



A Controller for Corrective Stepping for a Bipedal Exoskeleton Robot

Bachelor Thesis

BSc Biomedical Technology

Joost Verhoeven

Graduation Committee

Committee Chair and Daily Supervisor:	dr. ir. Arvid Keemink
Committee Member:	Ander Vallinas Prieto, MSc
External Member:	dr. Islam Khalil

Date:
03-07-2023

Document number:
BE-944

UNIVERSITY OF TWENTE.

Acknowledgments

I want to thank Ander Vallinas Prieto and dr. Islam Khalil for being a part of my graduation committee and for their feedback halfway through my assignment. I especially want to thank dr. ir. Arvid Keemink for being the committee chair and my daily supervisor. Our weekly meetings were inspiring and the mini-lectures I occasionally got during these meetings were valuable. Furthermore, I want to thank Arvid for spiking my interest in biorobotics during the minor BioRobotics, which lead me to this assignment. Lastly, I want to thank my friends and family for their support during my bachelor's degree.

Abstract

Up to half a million people worldwide suffer a spinal cord injury every year, often resulting in paraplegia. Exoskeleton robots can aid in rehabilitation and improve mobility and quality of life for these patients. Because current prototypes force the user to rely on crutches for stability, balance control is one of their main challenges. In this work, a controller for corrective stepping for a bipedal exoskeleton robot is designed and tested in simulations using a Linear inverted Pendulum (LiP) model. The controller uses the Instantaneous Capture point and Capture Areas associated with the LiP model to control the position of the feet and Zero Moment Point location within the Polygon of Support to stabilize the robot after a disturbance (e.g. a push). This controller proved to effectively stabilize pushes of up to 90 N in different directions for a duration of 0.25 s in simulations.

Keywords: Bipedal exoskeleton robot, balance control, corrective stepping, Linear inverted Pendulum (LiP), capturability

Contents

Chapter 1

Introduction

According to the World Health Organization (WHO), up to half a million people suffer a spinal cord injury (SCI) every year [1]. Most SCIs are caused by traffic accidents, falls, or violence. One major consequence of SCI is (partial) paraplegia, which means the patient (partially) loses control over the lower extremities. These patients often permanently end up in a wheelchair [2].

Acute care is important in SCIs. However, secondary complications also pose a significant threat, so careful aftercare and rehabilitation are required. Rehabilitation is also often necessary to reintroduce patients back into society and make them less dependent on caregivers. This can both improve the quality of life of the patients and reduce the costs of the disease, for example by reducing the lost earnings which is the main contributor to the costs [1].

Current rehabilitation methods rely on wheelchair usage and physical therapy [3]. However, this greatly reduces the mobility and independence of patients. New treatments in development are diverse, from neurological implants and prosthetics to exoskeleton robots [4]. These new treatments have the opportunity to improve the healing process and better assist paraplegics in daily life.

1.1 Exoskeleton Robots

Exoskeleton robots are robotic devices that can be worn over all or part of the human body (see Fig. 1.1). They can either be actuated or passive, and can provide support and assist in human locomotion. Exoskeletons therefore have the potential to improve the mobility and quality of life of SCI patients, assist in rehabilitation, and reduce the risk of secondary health complications, and results so far are promising [6, 7].

Still, there are some important challenges to overcome before exoskeletons for paraplegics can be deployed on a large scale [8]. One of those challenges is the balance control of these robotic devices. Many current prototypes, like the Symbitron Exoskeleton [5], have achieved gait support, but force the user to rely on crutches for balancing. However, this limits the mobility improvement the exoskeleton is supposed to provide. So, the exoskeleton should be able to balance on its own.

Fig. 1.1: The Symbitron Exoskeleton Robot. From [5].

1.2 Contribution of this Work

In recent work by A. Vallinas et al. [9], a momentum-based balance controller for the Symbitron robot during stance was developed. With this controller, the robot can withstand pushes of up to 30 N in forward and sideways directions. However, for higher disturbances or backward pushes, a recovery step is necessary, which the exoskeleton robot is currently not able to make. In this work, a controller for corrective stepping for a bipedal exoskeleton robot (like Symbitron) will be developed and discussed.

A Linear inverted Pendulum (LiP) model will be used to simulate a bipedal exoskeleton robot. With this model, a finite state machine controller will be developed and tested. The goal of the controller is to detect disturbances in the balance of the exoskeleton and take corrective steps based on these disturbances to stabilize the robot. Additionally, the controller must take limitations of the robot into account, like maximum step length and minimum step time, and it should avoid self-collision.

1.3 Outline of the Thesis

First, some background information regarding the Symbitron exoskeleton robot, balance, and the LiP model will be discussed. Next, the model used will be explained, as well as the designed controller and validation methods. After presenting the validation results, the controller and its behavior will be discussed, and possibilities for future work will be given.

Chapter 2

Background

In this chapter, first, some background information on the Symbitron exoskeleton will be given. Then, balance in bipedal robots will be discussed, before the LiP model and its features will be explained.

2.1 Symbitron

The Symbitron exoskeleton (Fig. 2.1 is a lower-limb exoskeleton (LLE) that can assist the user in standing and walking [5]. Its modular design allows it to be used in different configurations, from just ankle support to complete hip-knee-ankle support, which is the configuration of interest for this research. Furthermore, the exoskeleton was designed to be able to resize, so it can be tailored to its pilot.

The exoskeleton weighs around 41 kg and is underactuated with a total of 16 degrees of freedom (6 for the floating base, and 5 per leg). In both legs 4 of the 5 joints are actuated; hip ab-/adduction (HAA), hip flexion/extension (HFE), knee flexion/extension (KFE), and ankle plantar-/dorsi-flexion (ADP). Ankle in-/eversion (AIE) is also possible, but not actuated. Lateral or medial rotation of the hip is not possible, hip in-/eversion is possible but not actuated and will be disregarded in this research.

The exoskeleton joints are force-controlled through Series Elastic Actuators (SEAs). This facilitates compliant joints, which makes the LLE safer to use with a human pilot. Furthermore, it allows patients with partial SCIs to influence the exoskeleton, providing a more natural cooperation between the pilot and the machine. This also emphasizes the supporting function of the LLE and can aid in rehabilitation.

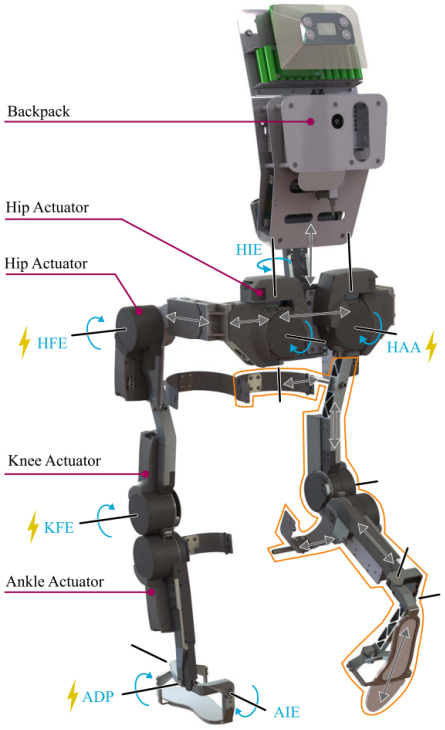


Fig. 2.1: A rendered CAD model of the Symbitron Exoskeleton Robot. From [5].

2.2 Stability of Bipedal Systems

In general, when talking about a stable controller, it means that for bounded input the output will also be bounded: BIBO stability. For the stability control of a bipedal robot, the input will be (bounded) disturbances to the stability of the system (like a push), and the output will either be that the exoskeleton stays upright (stable) or falls over (unstable). So, the controller can only be stable for a subset of disturbances (inputs), and other inputs like large disturbances that are physically impossible to withstand will lead to instability.

To avoid confusion and focus on the stability of the actual exoskeleton, two different terms for the stability of a bipedal robot are used in this work. The system is **stable** if it can stay standing upright without stepping, and it is **capturable** if it can become stable with a finite number of steps. In the following section, these definitions will be further specified.

2.3 The Linear inverted Pendulum as a Bipedal Robot Model

Bipedal (exoskeleton) robots are inherently unstable systems, which are difficult to control due to their high-order and non-linear dynamics. Furthermore, they are hybrid systems, because the dynamics change between single- and double-support phases. So, for the design of a controller for corrective stepping for a bipedal exoskeleton robot, a (greatly) simplified model is used: the Linear inverted Pendulum (LiP) model.

2.3.1 The Model

In the LiP model, first used for bipedal robot control in 1991 by S. Kajita and K. Tani [10], all mass is concentrated in the Center of Mass (CoM), $\mathbf{r}_{\text{CoM}}$, attached to a telescoping massless leg which pivots around the Center of Pressure (CoP), $\mathbf{r}_{\text{ZMP}}$. The rectangular area around the CoP represents the Polygon of Support (PoS), so the area underneath the (imaginary) feet of the robot wherein the CoP is physically able to be. The leg telescopes in such a way that the CoM always stays at the same height, h. The force it exerts on the CoM to accomplish this is represented by vector $\mathbf{f}$. See Fig. 2.2 for a schematic overview of the model.

The resultant force from the pressure between the feet of the robot and the ground has an equivalent Ground Reaction Force (GRF) that acts at the CoP, i.e. the point the LiP pivots around. If the GRF is inside the PoS, the CoP coincides with the Zero Moment Point (ZMP) [11]. The ZMP is the point where the CoP would need to be to ensure no rotational acceleration of the structure around any of the edges of the support. Whenever the ZMP moves outside the finite-sized foot, the GRF and thus CoP will lie on the edge of the foot, resulting in rotation of the robot around

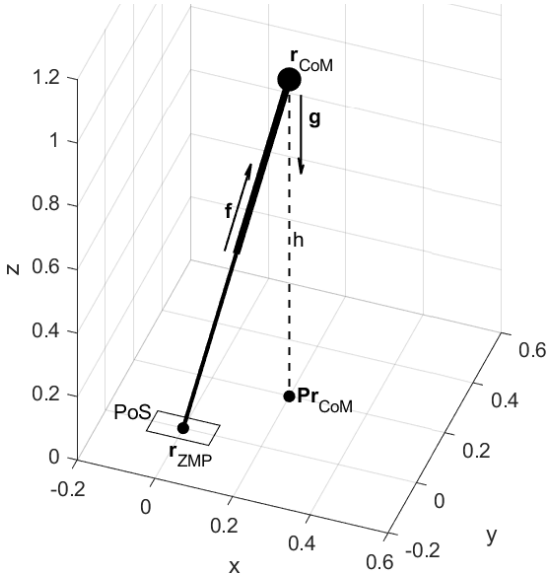


Fig. 2.2: A 3D-LiP model with a finite-sized foot. The foot, or PoS, is the rectangle around the base of the pendulum, the ZMP. The CoM pivots around the ZMP because of the gravity force $\mathbf{g}$*, but stays at the same height* h *through the force* $\mathbf{f}$ *exerted on the CoM by the telescoping leg. The point* $\mathbf{Pr}_{\text{CoM}}$ *is the location of the CoM projected onto the* xy*-plane. Forward is in the positive* x*-direction.*

that foot edge. Because of these features, the ZMP has been used by many researchers for the control of bipedal robots [11]. It can be used for gait generation but also for balancing, and as such it will be the input of the LiP model which will be controlled.

Using the LiP model, the hybrid, high-order, and non-linear dynamics of a bipedal robot are simplified to general, low-order, and linear dynamics. The equations of motion for this system are

$$m\ddot{\mathbf{r}}_{\text{CoM}}(t) = \mathbf{f}(t) + m\mathbf{g}, \tag{2.1}$$

where m is the mass at the top of the pendulum, $\mathbf{r}_{\text{CoM}} = \begin{pmatrix} x_{\text{CoM}} & y_{\text{CoM}} & z_{\text{CoM}} \end{pmatrix}^T$ is the position of the CoM, $\mathbf{f} = \begin{pmatrix} f_x & f_y & f_z \end{pmatrix}^T$ is the force acting on the CoM created by the telescoping leg, and $\mathbf{g} = \begin{pmatrix} 0 & 0 & -g \end{pmatrix}^T$ is the gravitational acceleration vector.

Using the location of the pivot point $\mathbf{r}_{\text{ZMP}} = \begin{pmatrix} x_{\text{ZMP}} & y_{\text{ZMP}} & z_{\text{ZMP}} \end{pmatrix}^T$ and the fact that the telescoping force cancels out the vertical motion/acceleration of the CoM, the equations of motion can be rewritten as

$$\ddot{\mathbf{r}}_{\text{CoM}}(t) = \frac{g}{h}\left(\mathbf{P}\mathbf{r}_{\text{CoM}}(t) - \mathbf{r}_{\text{ZMP}}(t)\right), \tag{2.2}$$

where $\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ projects $\mathbf{r}_{\text{CoM}}$ onto the xy-plane [12].

This results in a linear model. The equations are decoupled, so a 3D LiP model can be interpreted as a combination of two 2D LiP models, which further simplifies subsequent calculations.

2.3.2 Instantaneous Capture Point

One of the possibilities that this LiP model brings, is the ability to calculate the location of the Instantaneous Capture Point (IC or ICP), $\mathbf{r}_{\text{IC}}$. This is the point where the ZMP should be for the LiP to stay balanced, as can be seen in Fig. 2.3. So, balance can now be defined as $\|\mathbf{P}\mathbf{r}_{\text{CoM}}(t) - \mathbf{r}_{\text{ZMP}}(t)\| \to 0$ as $t \to \infty$, so the CoM stays above the pivot point/foot as time progresses.

By noting that Eq. (2.2) represents a mass-spring system, orbital energy equations can be derived, that can tell something about the rotation/pivoting of the system [14]. If this orbital energy is negative, the CoM will start falling backward before getting straight above the ZMP, and if it is positive the pendulum will topple over, as in Fig. 2.3 (left and right respectively). If the orbital energy is zero, the LiP model will come to a stop with the CoM above the ZMP and thus will be balanced. Using this, the IC point can be found [10, 12]. This results in the following equation:

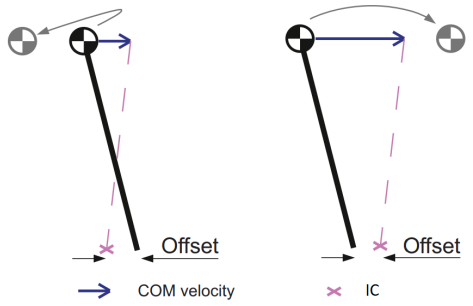


Fig. 2.3: An illustration of the Instantaneous Capture point (IC). If the ZMP is placed behind the IC, the pendulum will fall backward, so to where it came from (left figure). If it is placed in front of the ZMP, the CoM will fall past the ZMP, so forward (right figure). Adapted from [13].

$$\mathbf{r}_{\text{IC}}(t) = \mathbf{P}\mathbf{r}_{\text{CoM}}(t) + \sqrt{\frac{h}{g}}\dot{\mathbf{r}}_{\text{CoM}}(t), \tag{2.3}$$

where $\mathbf{r}_{\text{IC}} = \begin{pmatrix} x_{\text{IC}} & y_{\text{IC}} & z_{\text{IC}} \end{pmatrix}^T$ is the location of the IC. This point, which considers both the location and speed of the CoM relative to the ZMP, was also introduced as the extrapolated

Centre of Mass (exCoM or XcoM) by Hof et al. [15] in their studies into dynamic stability. Since the dynamics of the IC and CoM are stably coupled [16], it can be used to determine stability for a LiP model with a finite-sized foot, which represents the PoS of a real exoskeleton. Stability is now defined as $\mathbf{r}_{\text{IC}} \in \text{PoS}$. So, if the IC leaves the PoS, balance is lost and a step (so a change of the PoS) is necessary to regain balance.

Because of the linear dynamics of the LiP model and thus of the IC, it is also possible to calculate [12] the location of the IC after step time t_s:

$$\mathbf{r}_{\text{IC}}(t_s) = (\mathbf{r}_{\text{IC}}(0) - \mathbf{r}_{\text{ZMP}}(0))\, e^{\sqrt{\frac{g}{h}}t_s} + \mathbf{r}_{\text{ZMP}}(0) \tag{2.4}$$

This is the location where the ZMP should be after a step that takes t_s time for the system to come to a stop, given that the ZMP does not move during the step and there are no further disturbances. It can be used to plan recovery steps, check capturability, and calculate capture areas.

2.3.3 Capturability and Capture Areas

When the robot is falling, it is important to be able to determine if the system is capturable, or if it should enter a certain safety state to lower the risk of injuries/damages when a fall is unavoidable. If the robot is able to make a step that brings the IC inside the PoS, the system is considered 1-step capturable. The area around the IC(t_s) where the robot can step for this to happen is considered the 0-step capturability area, because after stepping there no more steps are required to stabilize the exoskeleton.

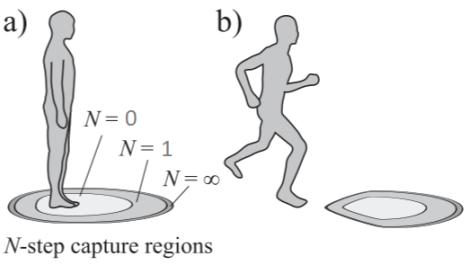


Fig. 2.4: A conceptual representation of N-step capturability areas for a human that is standing (a) and running (b). Adapted from [12].

Similar to the 0-step capturability area, higher N-step capturability areas can also be determined. So, if the exoskeleton robot is not able to step into the 0-step capturability area, it can be checked if the system is capturable in more than one step. See Fig. 2.4 for a representation of these N-step capturability areas for a standing and running human. For the running human, the IC(t_s) lies in front of the human and so do the capturability areas.

N-step capturability areas [12] are calculated using the following equation:

$$d_N = (d_{N-1} + l_{\max} - f_l)\, e^{-\sqrt{\frac{g}{h}}t_s} + f_l, \quad N \geq 1 \tag{2.5}$$

where $l_{\max}$ is the maximum step length and f_l is the average distance from the center of the foot to the edge of the PoS. The variable d_N is the radius of the circle around $\mathbf{r}_{\text{IC}}(t_s)$ wherein the ZMP should lie for the system to be N-step capturable.

As stated before, a corrective step is needed to regain stability if the IC moves outside the PoS. As a consequence, d_0 is (roughly) equal to the average distance between the center of the foot and the edge of the PoS, f_l. So, the parameters where d_N depends on are constant, which means the capturability areas are constant as well.

The series equation for d_N also allows the calculation of d_∞ [12]:

$$d_\infty = l_{\max} \frac{e^{-\sqrt{\frac{g}{h}}t_s}}{1 - e^{-\sqrt{\frac{g}{h}}t_s}} + f_l \tag{2.6}$$

So, if the model/robot is not able to step within the d_∞ area (i.e. the distance from both feet to $\mathbf{r}_{\text{IC}}(t_s)$ is larger than $l_{\max} + d_\infty$), the system is not capturable and a safe falling strategy should be adopted. The latter is outside of the scope of this work.

Chapter 3

Methods

For the design of a controller for corrective stepping in a bipedal exoskeleton robot, a 3D-LiP model was used. First, this model is discussed, before the controller itself is explained. Finally, the implementation and controller validation methods are stated.

3.1 The Model

A 3D-LiP model as discussed in Chapter 2, Sec. 2.3 was used to simulate a bipedal exoskeleton. In this model, the location of the base of the pendulum, thus the ZMP, is the input, and the location and velocity of the CoM are the output. In addition, two virtual massless feet connected to the CoM with massless legs were added, so the model more closely resembles an actual bipedal robot and individual foot placement can be taken into account.

Table 3.1: Model parameters

Parameter	Value	Description
m	41 kg	Mass at top of pendulum
h	1 m	Height of pendulum
g	9.81 m s^{-2}	Gravitation constant
w	0.2 m	Hip/stance width
$l_{\max}$	0.4 m	Maximum step length
t_s	0.5 s	Step time
t_{ws}	0.12 s	Weight shifting time before step
t_{cd}	0.05 s	Cooldown time after step
f_w	0.04 m	Foot width from center of foot
f_h	0.08 m	Foot height from center of foot

Table 3.1 shows the parameters that are used for this model. The hip/stance width is the distance between the centers of the feet during stance. The maximum step length is the maximum distance a foot can step, measured from the other foot. Apart from the stepping time t_s, two other time constants are used: weight shifting time before a step t_{ws}, which helps counteract falling sideways when stepping forward (will be explained in more detail in Sec. 3.2), and cooldown time after a step t_{cd}, which allows for the shifting of weight (and thus movement of the ZMP) to the other foot after a step. The foot width and height f_w and f_h are not necessarily the actual dimensions of the robot's feet, but rather the area within the feet where

the ZMP can lie. When both feet are on the ground, the ZMP is able to lie within the convex hull of the feet.

Using these parameters, the capturability areas were calculated. The distance f_l was approximated by averaging the foot width and height. Furthermore, t_s in Eq. (2.5) and (2.6) was substituted with $t_s + t_{cd}$, to account for the added cooldown time to the step time. The following radii (in m) were obtained: $[d_0 = 0.0600, d_1 = 0.1314, d_2 = 0.1442, d_3 = 0.1465, ..., d_\infty = 0.1470]$. After d_2 the difference in radius becomes insignificant because the robot will not be able to step that precisely. Therefore, 3-step capturability areas, and higher, will be disregarded.

3.2 The Controller

To be able to effectively stabilize and/or capture the system, the controller is required to be able to adapt its behavior to suit the current situation. To this end, a finite state machine was designed and implemented for the control of the model. It was designed to take the location and velocity of the CoM and the location of the left and right foot of the robot as inputs, and calculate the desired position of the ZMP and left and right foot as outputs. Furthermore, it was designed to prevent collision between the two legs, weight shifting was added, and a capturability check was implemented.

3.2.1 The State Machine

Fig. 3.1 shows a diagram of the state machine that was designed. The controller starts on simulation start and enters the `Standing` state. In this state, it continuously updates the IC, ZMP, and IC(t_s) based on the position and velocity of the CoM. Whenever a disturbance (like a push) is administered to the CoM, the controller first tries to correct for it by moving the ZMP accordingly, so by shifting the weight of the robot. However, when the IC no longer lies within the PoS and the exoskeleton thus lost its balance, this tactic is no longer sufficient and the controller switches to the `Falling` state.

In the `Falling` state, the IC, ZMP, and IC(t_s) are again updated. Then, the capturability of the system is checked. If the distance between both feet and $\mathbf{r}_{\text{IC}}(t_s)$ is larger than the maximum step length $l_{\max}$ plus the 2-step capturability area radius d_2, the system is considered uncapturable and the controller stops. When the system is capturable, a decision with which leg to step is made. If the exoskeleton currently has its legs crossed, it will step with the left leg if the IC(t_s) lies to the left of the robot and vice versa. Otherwise, it will check if one of the feet is supporting more than 80% of the weight of the robot and, if so, choose to step with the other foot. Finally, if that is also not the case, the controller checks which foot is closer to the IC(t_s) and if it is within stepping range. If so, it will choose that foot to step with, otherwise, the other foot will be able to step closer to the IC(t_s) and thus will be chosen.

If the robot was not standing with its legs crossed and did not have more than 80% of its weight on one foot, the state machine first enters a `Weight Shifting` state before the actual stepping state. In this state, the ZMP is first moved to the foot that will perform the step. This weight shifting is done to counteract the sideways falling of the robot during the step, caused by the movement of the ZMP from between the feet to the foot that is not stepping. After t_{ws} seconds have passed, the ZMP moves to the other foot and the controller switches to the corresponding `Stepping` state.

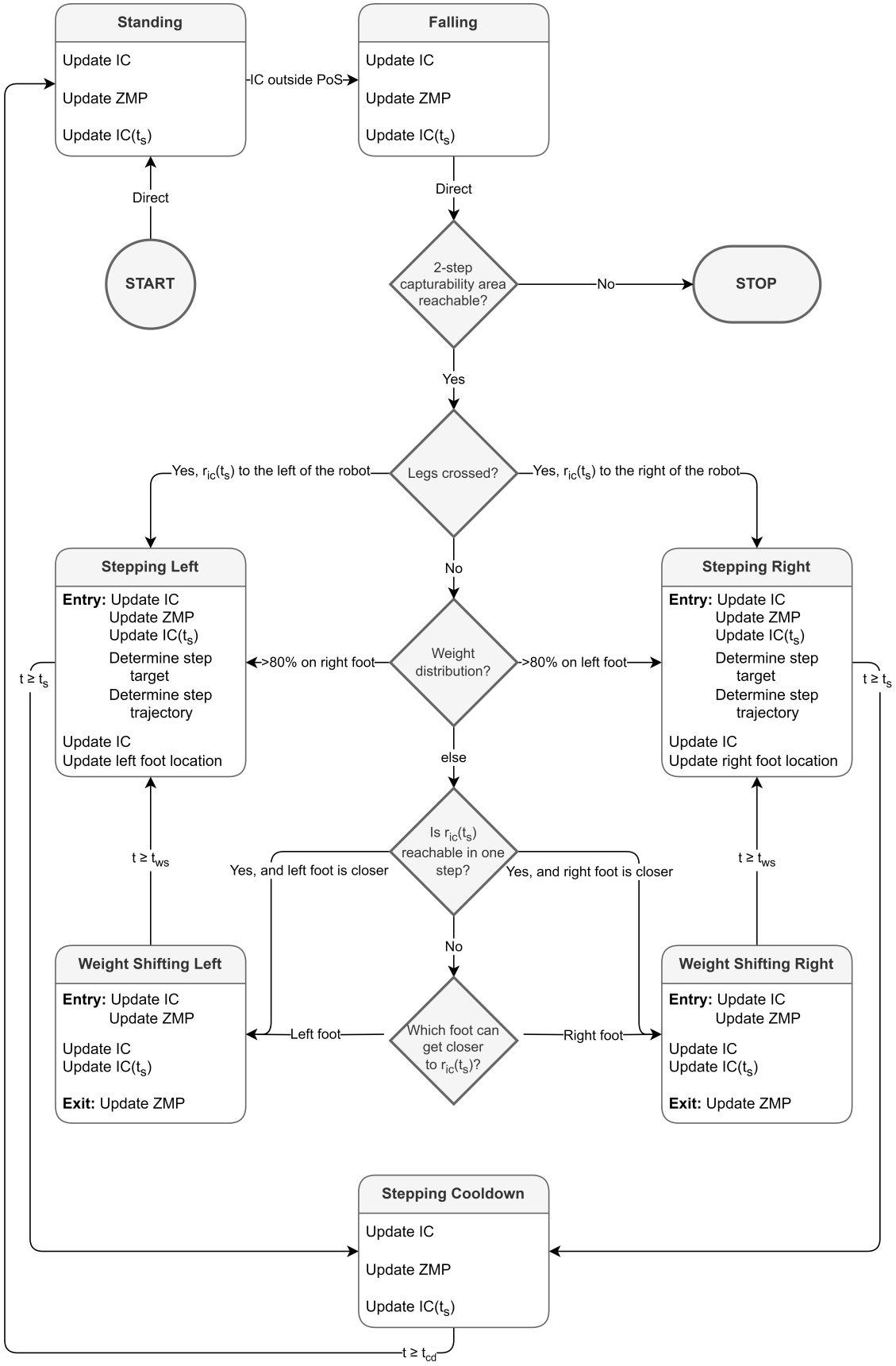


Fig. 3.1: Overview of the controller, containing a flow diagram of the finite state machine and algorithms used.

In the `Stepping` state, first, the IC, ZMP, and IC(t_s) are updated. Then, the step target is determined. A vector from the non-stepping foot towards the $\mathbf{r}_{\text{IC}}(t_s)$ is formed, and extended by d_0. This extension helps increase the robustness of the controller by increasing the stability margin. The stability margin is defined as the distance from the ZMP to the edge of the PoS, as this is a measurement of the amount of movement of the ZMP that is allowed to keep the system stable. If this vector is greater than $l_{\max}$, it is scaled down to take the maximum step length into account. Finally, the controller checks if the step target causes no collision between the feet, and, if stepping with the right foot, the location is not to the left of the left foot and vice versa. This helps prevent internal collisions and facilitates a more natural way of crossed legs stepping to the right or left. Using the now-found step target, a stepping trajectory is determined. Then, during t_s seconds, the IC is updated and the foot is made to follow the trajectory, before ending at the step target.

The controller now switches to the `Stepping Cooldown` state. Here, the IC, ZMP, and IC(t_s) are updated for a duration of t_{cd} seconds. This allows for the movement of the ZMP and thus weight shifting from the foot that did not step to the new ideal ZMP location in the new PoS. The controller now switches back to the `Standing` state, where it will check if the robot regained stability or if it is still falling and another step is required.

3.2.2 Desired ZMP Location and Stepping Trajectory

Whenever the ZMP is updated, the controller first checks if the IC lies within the PoS. If it does, the ZMP is set to this location. When the IC lies outside the PoS, however, it searches for the closest point within the PoS to the IC. It does this by first finding the edges of the PoS, which is a rectangle when only one foot is on the ground, and a convex hull with 6 corners if both feet are on the ground. It then loops through the edges and finds the closest point to the IC on each edge. Finally, from the points found, the one closest to the IC is chosen. See Appendix A.1 for a more detailed explanation.

For the stepping trajectory, three 6th-order polynomials are used (one for every dimension), which start at the current foot position with a velocity and acceleration of zero, and end at the step target, also with zero velocity and acceleration. Halfway through the step, the polynomials are designed to pass through a point halfway in between the starting and ending point, but with a greater z-value to create an arc. The height of the arc is determined by the stepping length, with a minimum of 5 cm and a maximum of 30 cm. See Appendix A.2 for a more detailed explanation.

3.3 Implementation

The 3D LiP model was implemented in Simulink (Version 10.5) in combination with MATLAB (R2022a). The controller was implemented as a Matlab Function within Simulink. For the simulation, a discrete fixed-step solver was used, with a sample time of 0.001 s, so 1000 Hz. A simulation time of 5 s was chosen, and it was checked if this time was sufficient by looking at the simulations with the most steps and if in these simulations the CoM came to a stop.

3.4 Controller Validation

A disturbance in the CoM's acceleration, $F_d = \begin{pmatrix} F_{d,x} & F_{d,y} & 0 \end{pmatrix}^T$, was added to simulate a push. This disturbance was set to start 0.1 s after the start of the simulation and end 0.25 s later at $t = 0.35$ s. The direction of the push was varied within the xy-plane, and the magnitude

of the push was varied from 0 to 100 N to test the controller. The resulting movements of the CoM, ZMP, IC, IC(t_s), and feet were recorded, plotted, and analyzed.

In particular, first pushes in forward, diagonal and sideways directions that result in a single correction step are analyzed. Plots of the situation at the beginning of the step, halfway through and just after the step are created, to show the step trajectory and robot movement. Furthermore, top-view trace plots are created, which show the movement of the CoM, IC, and ZMP, and the foot positions.

A stronger push in forwards and sideways directions which the controller is still able to stabilize is also plotted and analyzed in the same way. Here, the behavior of the controller during multiple steps is more closely studied.

Finally, the controller is tested for pushes with $F_{d,x} \in [-80, 80]$ N (steps of 20 N) and $F_{d,y} \in [-100, 100]$ N (also steps of 20 N). A contour plot is created, showing the number of steps required to regain stability. If the system is not capturable, nothing is plotted. This will help determine the limits of the controller. Any special cases will be noted and analyzed further.

Chapter 4

Results

The results from the controller validation will be presented in this chapter. The reaction of the controller to different disturbances will be shown, and particularities will be highlighted.

4.1 Pushes in Forward, Diagonal, and Sideways Directions

In Figs. 4.1–4.3 the results of the simulations for respectively forward, diagonal, and sideways 1-step capturable pushes are shown. The situation at the beginning of the step, halfway through, and after the step is depicted. Note that the IC(t_s) is not updated during the step, but is updated afterward, so its location and thus the location of the capturability areas in the third subplot can differ from the first two in each figure.

Due to the forward push, $F_d = \begin{pmatrix} 60 & 0 & 0 \end{pmatrix}^T$ N, a step is made with the right foot (Fig. 4.1). The IC, IC(t_s), and capturability areas lie slightly to the right of the robot. It is also visible that the step target is past the IC(t_s), not by d_0 but on the maximum step distance from the left foot. After the step, the ZMP is moved from the left foot to the IC that lies in the right foot.

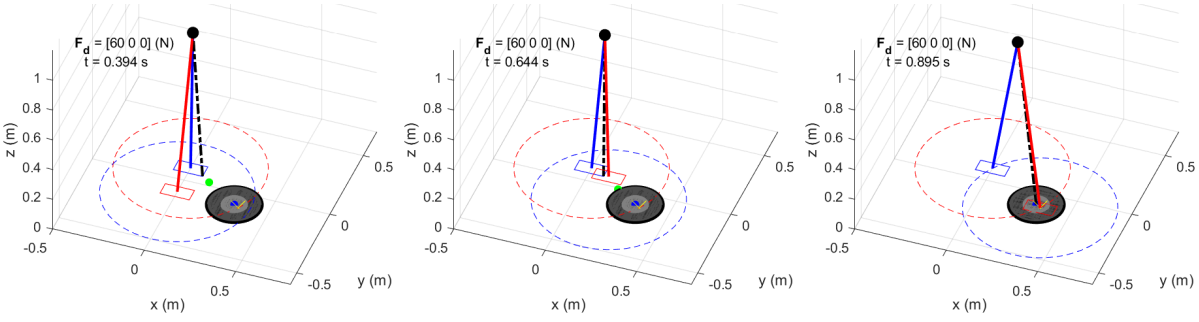


Fig. 4.1: Simulation of model and controller response after a forwards push of $F_d = \begin{pmatrix} 60 & 0 & 0 \end{pmatrix}^T$ N. The LiP model can be seen as the dash-dotted black line which connects the ZMP on the ground to the CoM (black circle). The blue and red rectangles represent the left and right foot respectively, and are connected to the CoM with the correspondingly colored line. The dashed red circle around the left foot shows the maximum stepping distance for the right foot, and vice versa for the dashed blue circle. The green dot shows the IC and the blue dot is the IC(t_s). Around the IC(t_s) the 0, 1, and 2-step capturability areas are shown, from light to dark gray. The yellow cross shows the step target. In the top left, the administered push and timestamp are shown.

Due to the diagonal push, $F_d = \begin{pmatrix} 45 & 45 & 0 \end{pmatrix}^T$ N, again a step is made with the right foot (Fig. 4.2). It is visible that the step target lies a bit more than d_0 past the IC(t_s), and that

the feet do not collide after the step, although they are very close together. The ZMP moves outside the PoS of the individual feet but stays inside the overall PoS (the convex hull of both feet).

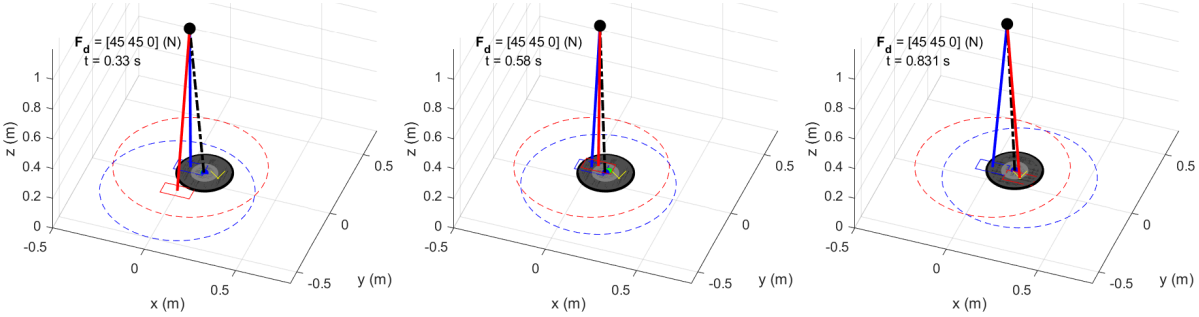


Fig. 4.2: Simulation of model and controller response after a diagonal push of $F_d = \begin{pmatrix} 45 & 45 & 0 \end{pmatrix}^T$ *N. The LiP model can be seen as the dash-dotted black line which connects the ZMP on the ground to the CoM (black circle). The blue and red rectangles represent the left and right foot respectively, and are connected to the CoM with the correspondingly colored line. The dashed red circle around the left foot shows the maximum stepping distance for the right foot, and vice versa for the dashed blue circle. The green dot shows the IC and the blue dot is the IC(t_s). Around the IC(t_s) the 0, 1, and 2-step capturability areas are shown, from light to dark gray. The yellow cross shows the step target. In the top left, the administered push and timestamp are shown.*

Due to the sideways push, $F_d = \begin{pmatrix} 0 & 75 & 0 \end{pmatrix}^T$ N, a step is made with the right foot again (Fig. 4.3). It can be seen that the right foot is not placed to the direct left of the left foot, but rather in front and to the left. The IC lies inside the PoS after the step, and the ZMP is moved there. The simulation ends with the exoskeleton in a stance with its legs crossed.

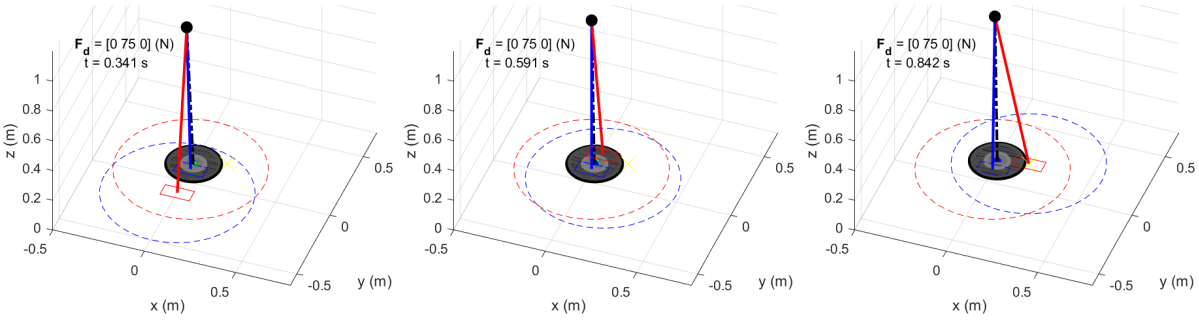


Fig. 4.3: Simulation of model and controller response after a sideways push of $F_d = \begin{pmatrix} 0 & 75 & 0 \end{pmatrix}^T$ *N. The LiP model can be seen as the dash-dotted black line which connects the ZMP on the ground to the CoM (black circle). The blue and red rectangles represent the left and right foot respectively, and are connected to the CoM with the correspondingly colored line. The dashed red circle around the left foot shows the maximum stepping distance for the right foot, and vice versa for the dashed blue circle. The green dot shows the IC and the blue dot is the IC(t_s). Around the IC(t_s) the 0, 1, and 2-step capturability areas are shown, from light to dark gray. The yellow cross shows the step target. In the top left, the administered push and timestamp are shown.*

In Fig. 4.4 top-down view trace plots of the model after the three different pushes can be seen. The blue and red rectangles again represent the left and right foot respectively. Additionally, the movement of the CoM, IC, and ZMP are shown. What stands out is that in the graph of the forward push the CoM and IC not only move forward (in positive x-direction) but also sideways. They first move a bit to the left (positive y), and then more towards the right

(negative y) before ending inside the PoS of the right foot. It can also be seen that, although the push was completely in the forward direction, the robot did not purely step forward with its right foot but also a bit to the right.

In the trace plot of the diagonal push simulation (middle figure), it can also again be seen that the controller prevented a collision between the feet. In the trace plot of the sideways push (right figure), it is also again visible that the right foot did not step to the direct left of the left foot, but rather to the left and in front of the left foot.

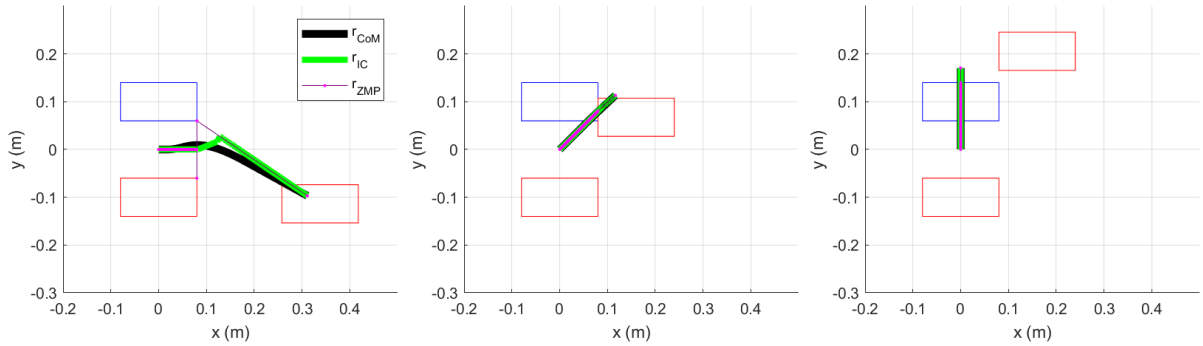


Fig. 4.4: Top view trace plots of the model after respectively forwards (left figure), diagonal (middle figure), and sideways (right figure) pushes. The blue and red rectangles show the stepping locations of the left and right foot respectively, including their starting position.

Only pushes in positive x- and y-directions are shown here. Pushes in other directions were also tested and showed similar results, and are therefore omitted.

4.2 Stabilizing with More Steps

Apart from the before mentioned 1-step capturable pushes in forward, diagonal, and sideways directions, pushes that required multiple steps to stabilize were also tested. In Fig. 4.5 the top view trace plots of two simulations can be seen, one with a stronger push in the forward direction $F_d = \begin{pmatrix} 80 & 0 & 0 \end{pmatrix}^T$ N and one in sideways direction $F_d = \begin{pmatrix} 0 & 85 & 0 \end{pmatrix}^T$ N.

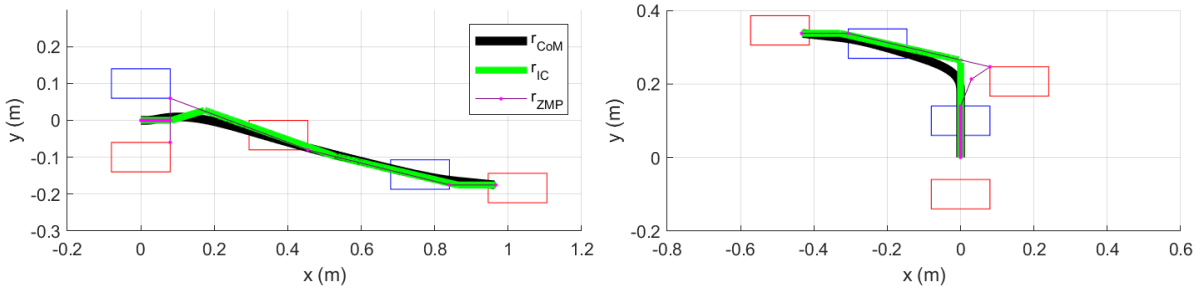


Fig. 4.5: Top view trace plots of the model after forward and sideways pushes that require multiple steps to capture. The blue and red rectangles show the stepping locations of the left and right foot respectively, including their starting position.

It can be seen that both pushes required three steps to stabilize. For the forwards push, it stands out that the first step with the right leg is a bit to the left, while the robot is falling a bit to the right. The weight shifting before this first step is also visible, the ZMP first moves to the right foot before going to the left foot to allow the stepping with the right foot. After the first step, the exoskeleton steps diagonally to the front right with the left foot, causing the legs to be crossed. Finally, a last step is made with the right foot, stabilizing the robot.

What is noticeable for the sideways push is that the first step with the right foot is to the front and left of the left foot, instead of directly left. After this first step, the robot starts also falling backward, so the following steps are diagonal to the left and back. The robot regains stability in a crossed-leg stance because the right foot is to the left of the left foot.

4.3 Capturability Overview

To test the limits and performance of the controller pushes of different magnitudes and directions were administered to the model ($n = 399$). Fig. 4.6 shows a filled contour plot on how many steps the robot needed to come to a stop if this was possible.

The light gray area in the middle contains all pushes that did not require a step, so these pushes were not strong enough to destabilize the robot. Then, in a bit darker gray the 1-step capturable pushes are visible, followed by the 2-step capturable pushes in even darker gray. Although there were pushes that required more steps to capture, there are too few to be visible in the contour plot.

What stands out is the relative sizes of the different N-step capturability areas. A large number of pushes (135) were 0-step capturable, while only 26 pushes were 1-step capturable, 38 pushes were 2-step capturable, and 12 steps were 3-step capturable. So, two capturing steps were also more often required than just one step. Remarkably, there were also two pushes which required five steps to stabilize. The other 186 pushes were not capturable.

To further analyze the 5-step capturable push case $F_d = \begin{pmatrix} 50 & -90 & 0 \end{pmatrix}^T$ N, a top view trace plot was created which can be seen in Fig. 4.7. It can be seen that the robot first crosses its legs by stepping to the front right of the right foot with its left foot. Afterward, it makes a large step (maximum step length) to the right with its right foot. It then cross-steps again, causing the exoskeleton to start falling backward. The next step is with the right foot, backward and to the right, and lastly, the left foot is used to step further back and stabilize the system. The other 5-step capture case is for the same disturbance but in negative $F_{d,x}$-direction, and looks very similar (but mirrored) to this case.

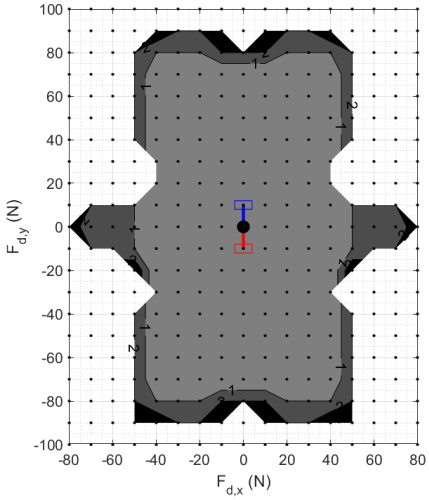


Fig. 4.6: A filled contour plot on the number of steps needed by the system to come to a stop after a push. A representation of the robot is added in the middle for orientation clarification. The robot faces to the right of the graph (positive $F_{d,x}$ direction).

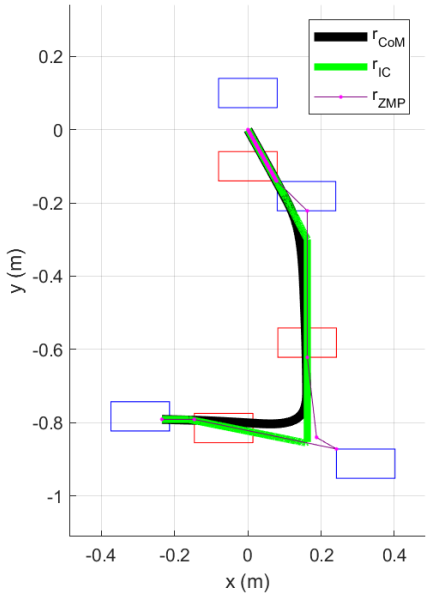


Fig. 4.7: Top view trace plot of simulation with five recovery steps. The blue and red rectangles show the stepping locations of the left and right foot respectively, including their starting position.

Chapter 5

Discussion

The goal of this research was to develop a controller for corrective stepping for a bipedal exoskeleton robot. Using a 3D LiP model and the associated IC, IC(t_s), and capturability areas, a finite state machine was designed that controls the location of the ZMP and feet of the exoskeleton. This controller proved to effectively stabilize pushes of up to 90 N in different directions for a duration of 0.25 s in simulations.

5.1 Performance

As can be seen in Chapter 4, the controller effectively performs capturing steps. The controller detects when there is a disturbance in the balance of the robot by calculating the location of the IC and checking if it lies inside the PoS. Whenever the IC moves outside the PoS, the controller plans a corrective step, taking the additional constraints into account. It limits the step distance to the maximum step length and sets the duration of the step to the minimal step time. It avoids self-collision by updating the step target so the feet do not overlap after the step, and by assuring the left foot is never placed directly to the right of the right foot and vice versa. The controller then effectively performs the corrective step.

Additionally, the controller's choice of foot to step with is based on the current stance and weight distribution. This helps generate more effective and natural capture steps. Furthermore, when the robot's legs are not crossed and no more than 80% of the total weight is on one foot, a weight-shifting trajectory of the ZMP is traced, as can be seen in Figs. 4.4 and 4.5. This weight shifting helps cancel out sideways falling caused by stepping with one foot which moves the ZMP towards the other foot, so towards the side.

5.2 Comparison with State of the Art

Comparing the controller developed in this work with other state-of-the-art controllers is difficult since no simulations or experiments with actual bipedal (exoskeleton) robots were performed. However, in simulations with a simple planar biped recovering balance after a push by stepping based on the IC [14], the controller recovered from a push resulting in a change of velocity of the CoM from rest to 0.2 m/s, which is relatively a smaller disturbance than the disturbances the controller from this work can handle. The LiP model used in [14] does, however, include a flywheel at the CoM and feet with mass, and only uses one recovery step, which makes a direct comparison less appropriate.

Recovery steps can also be based on other characteristics than the IC, like the Divergent Component of Motion (DCM) [17]. The humanoid robot Toro can withstand lateral pushes of

25 N for 1 s while walking, and can survive external forces of up to 15% of the robot's weight (76 kg) being controlled based on the DCM. Since the controller in this work focuses on balance recovery during stance, a direct comparison cannot be made, however, the performances of the controllers seem similar.

Disturbance recovery of humanoid robots can also be based on model predictive control (MPC) and whole-body dynamics [18]. Using these strategies, a humanoid robot (41 kg) was able to withstand a forward push of 400 N for 0.1 s in simulations, which is a (relatively) better performance than that of the controller developed in this work. However, this robot also uses e.g. movement of the arms for recovery, which is not possible for exoskeleton robots like Symbitron.

5.3 Limitations and Possible Improvements

For forward pushes it can still be seen that the exoskeleton also falls or moves a bit to the side caused by the displacement of the ZMP to the foot which is not stepping (see Figs. 4.4 (left) and 4.5 (left)). This indicates that the weight shifting before the step is not ideal and that this still provides an opportunity for improvement. It could be useful to implement a dynamic weight-shifting trajectory or time instead of the static ones currently used, which adapt to the situation by taking the positions of the feet and position and velocity of the CoM into account and better cancel out sideways movements.

Furthermore, it may also be useful to perform weight shifting in other situations, like the sideways push visible on the right in Fig. 4.5. It can be seen that although the push was directed to the left, the robot also moves back, due to stepping in front of the left foot instead of directly to the left. So, a front-back weight-shifting might (partially) cancel this movement, possibly leading to faster recovery and increased capturability.

Apart from the possibility of dynamic weight shifting, the stepping itself can also benefit from being more dynamic [19]. Currently, every step takes the same amount of time, which limits the possibilities of the controller and will also feel unnatural for the pilot for small steps versus large steps. So, an adaptive stepping time might be preferable over a static one. Furthermore, updating the step target and trajectory during the step can improve accuracy, and better handle disturbances during the step.

The controller successfully prevents collisions between the feet by moving the step target. However, the stepping trajectory does not take this constraint into account, and the stepping foot is free to move through the other foot or leg during the step. So, before the controller can be tested on a real exoskeleton, this constraint must also be taken into account when generating the stepping trajectory. Additionally, the controller now only moves the feet and sees the legs as virtual connections between the feet and CoM. The legs of the real robot are of course not virtual, so trajectories should also be created for them.

What also stands out is the sizes of the N-step capture regions in Fig. 4.6. Many pushes turned out to be 0-step capturable, while the amount of 1-step capturable pushes was low. As can be seen in [9], the actual 0-step capturability region should be smaller for the Symbitron exoskeleton. Further tweaking of the model parameters might improve the accuracy of the simulation.

In addition, in the simulations in this research, the actual ZMP of the model instantaneously moves towards the desired ZMP location, as determined by the controller. With a real exoskeleton, this is not possible, there would be a (small) delay before the ZMP reaches the desired location, due to e.g. actuator dynamics. This delay can be integrated into the model with for example a low-pass filter to better resemble the actual robot [16].

5.4 Future Work

Apart from the before mentioned possible improvements on limitations of the controller, there are some additional extensions/adaptions the controller might benefit from. One of these proposed extensions to the controller is an extra control sequence that brings the exoskeleton back to a parallel stance after stabilizing, so with the feet next to each other at hip/stance distance w. This increases the stability margin, providing more stability. Furthermore, it is desired for the pilot to return to a more natural stance after stabilizing.

Other ways of improving push robustness are implementing more advanced models, like the Variable-Height Inverted Pendulum (VHIP) [20], Augmented Linear Inverted Pendulum (ALIP) [21], or Bipedal Trunk Spring Loaded Inverted Pendulum (BTSLIP) [22] model. These models more closely resemble the dynamics of an actual bipedal exoskeleton robot and provide more control possibilities to increase stability. Apart from these more advanced models the implementation of rotational inertia into the model will also increase simulation accuracy, and can also improve capturability [12].

Lastly, the controller is at this point only able to walk on flat ground. In daily life, SCI patients using an exoskeleton robot will also have to walk on uneven terrain, so the controller should be adapted to also be able to balance the exoskeleton under these (more challenging) circumstances, including varying foot placement heights and ground frictions.

References

1. World Health Organization. Spinal cord injury. 2013. Date accessed: 26/05/2023. Available from: https://www.who.int/news-room/fact-sheets/detail/spinal-cord-injury
2. National Institute of Neurological Disorders and Stroke. Spinal Cord Injury. 2023. Date accessed: 30/05/2023. Available from: https://www.ninds.nih.gov/health-information/disorders/spinal-cord-injury
3. Ozelie R, Sipple C, Foy T, Cantoni K, Kellogg K, Lookingbill J, Backus D, and Gassaway J. SCIRehab Project Series: The Occupational Therapy Taxonomy. The Journal of Spinal Cord Medicine 2009; 32:283
4. Failli V, Kleitman N, Lammertse DP, Hsieh JT, Steeves JD, Fawcett JW, Tuszynski MH, Curt A, Fehlings MG, Guest JD, and Blight AR. Experimental Treatments for Spinal Cord Injury: What you Should Know. Topics in Spinal Cord Injury Rehabilitation 2021 Mar; 27:50
5. Meijneke C, Van Oort G, Sluiter V, Van Asseldonk E, Tagliamonte NL, Tamburella F, Pisotta I, Masciullo M, Arquilla M, Molinari M, Wu AR, Dzeladini F, Ijspeert AJ, and Van Der Kooij H. Symbitron Exoskeleton: Design, Control, and Evaluation of a Modular Exoskeleton for Incomplete and Complete Spinal Cord Injured Individuals. IEEE Transactions on Neural Systems and Rehabilitation Engineering 2021; 29:330–9
6. Strausser KA, Swift TA, Zoss AB, and Kazerooni H. Prototype Medical Exoskeleton for Paraplegic Mobility: First Experimental Results. ASME 2010 Dynamic Systems and Control Conference, DSCC2010 2011 Jan; 1:453–8
7. Baunsgaard CB, Nissen UV, Brust AK, Frotzler A, Ribeill C, Kalke YB, León N, Gómez B, Samuelsson K, Antepohl W, Holmström U, Marklund N, Glott T, Opheim A, Murillo N, Nachtegaal J, Faber W, Biering-SØrensen F, and Benito Penalva J. Exoskeleton gait training after spinal cord injury: An exploratory study on secondary health conditions. Journal of rehabilitation medicine 2018; 50:806–13
8. Gorgey AS. Robotic exoskeletons: The current pros and cons. World Journal of Orthopedics 2018 Sep; 9:112
9. Vallinas A, Keemink A, Bayón C, Van Asseldonk E, and Van Der Kooij H. Momentum-Based Balance Control of a Lower-Limb Exoskeleton During Stance. To be published in Proceedings of ICORR 2023
10. Kajita S and Tani K. Study of dynamic biped locomotion on rugged terrain–Derivation and application of the linear inverted pendulum mode. Proceedings - IEEE International Conference on Robotics and Automation 1991; 2:1405–11
11. Vukobratovic M and Borovac B. Zero-Moment Point - Thirty Five Years of its Life. International Journal of Humanoid Robotics 2004 Mar; 1:157–73
12. Koolen T, De Boer T, Rebula J, Goswami A, and Pratt J. Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models. The International Journal of Robotics Research 2012 Jul; 31:1094–113
13. Vlutters M, Van Asseldonk EHF, and Van Der Kooij H. Center of mass velocity-based predictions in balance recovery following pelvis perturbations during human walking. Journal of Experimental Biology 2016 :1514–23

14. Pratt J, Carff J, Drakunov S, and Goswami A. Capture point: A step toward humanoid push recovery. Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS 2006 :200–7

15. Hof AL, Gazendam MG, and Sinke WE. The condition for dynamic stability. Journal of Biomechanics 2005 Jan; 38:1–8

16. Englsberger J, Ott C, Roa MA, Albu-Schaffer A, and Hirzinger G. Bipedal walking control based on Capture Point dynamics. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems 2011 Dec :4420–7

17. Englsberger J, Mesesan G, and Ott C. Smooth trajectory generation and push-recovery based on Divergent Component of Motion. IEEE International Conference on Intelligent Robots and Systems 2017 Dec; 2017-September:4560–7

18. Shi X, Gao J, Lu Y, Tian D, and Liu Y. Simulation of disturbance recovery based on mpc and whole-body dynamics control of biped walking. Sensors (Switzerland) 2020 May; 20

19. Kim G, Kuribayashi H, Tazaki Y, and Yokokohji Y. Omni-Directional Fall Avoidance of Bipedal Robots with Variable Stride Length and Step Duration. IEEE-RAS International Conference on Humanoid Robots 2019 Jan; 2018-November:718–24

20. Yang S, Chen H, Zhang L, Cao Z, Wensing PM, Liu Y, Pang J, and Zhang W. Reachability-based Push Recovery for Humanoid Robots with Variable-Height Inverted Pendulum. Proceedings - IEEE International Conference on Robotics and Automation 2021; 2021-May:9022–8

21. Dau H, Chew CM, and Poo AN. Proposal of augmented linear inverted pendulum model for bipedal gait planning. IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings 2010 :172–7

22. Vu MN, Lee J, and Oh Y. Control strategy for stabilization of the biped trunk-SLIP walking model. 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2017 2017 Jul :1–6

Appendix A

Algorithms

In this appendix the used algorithms will be explained in more detail.

A.1 Desired ZMP Location

To find the closest point in the PoS when the IC lies outside the PoS, first, all corners of the PoS are found. Then, the controller loops through these points (except the first one) and finds the point on the line segment connecting each point with the previous point that is the closest to the IC. For points $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{r}_{\text{IC}}$ this is done by first creating a vector from point 1 to point 2, and from point 1 to the IC:

$$\overrightarrow{\mathbf{p}_1\mathbf{p}_2} = \mathbf{p}_2 - \mathbf{p}_1, \tag{A.1}$$

$$\overrightarrow{\mathbf{p}_1\mathbf{r}_{\text{IC}}} = \mathbf{r}_{\text{IC}} - \mathbf{p}_1. \tag{A.2}$$

The IC is projected on the line passing through $\mathbf{p}_1$ and $\mathbf{p}_2$ with

$$d = \frac{\overrightarrow{\mathbf{p}_1\mathbf{p}_2}}{\|\overrightarrow{\mathbf{p}_1\mathbf{p}_2}\|} \cdot \overrightarrow{\mathbf{p}_1\mathbf{r}_{\text{IC}}} \tag{A.3}$$

$$\mathbf{r}_{\text{IC,proj}} = \mathbf{p}_1 + d\frac{\overrightarrow{\mathbf{p}_1\mathbf{p}_2}}{\|\overrightarrow{\mathbf{p}_1\mathbf{p}_2}\|}. \tag{A.4}$$

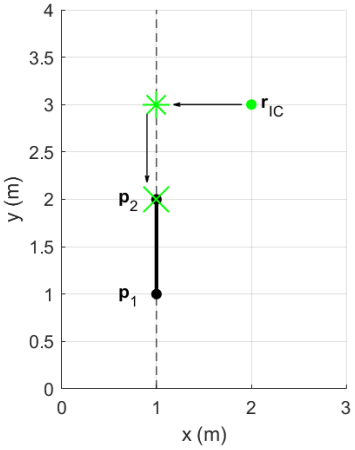


Fig. A.1: Finding the closest point on line segment $\mathbf{p}_1\mathbf{p}_2$ to point $\mathbf{r}_{\text{IC}}$.

However, to limit the projection to the line segment, d is first limited between 0 and the length of the vector $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$.

Repeating this step for all corners of the PoS and thus for all edges of the PoS, a set of possible points is found. The final point is then chosen by calculating the distance between each point and the IC and choosing the point with the smallest distance.

A.2 Stepping Trajectory

The 6th-order polynomials used for the stepping trajectory are:

$$x(t) = a_{x,0} + a_{x,1}t + a_{x,2}t^2 + a_{x,3}t^3 + a_{x,4}t^4 + a_{x,5}t^5 + a_{x,6}t^6, \tag{A.5}$$
$$y(t) = a_{y,0} + a_{y,1}t + a_{y,2}t^2 + a_{y,3}t^3 + a_{y,4}t^4 + a_{y,5}t^5 + a_{y,6}t^6, \tag{A.6}$$
$$z(t) = a_{z,0} + a_{z,1}t + a_{z,2}t^2 + a_{z,3}t^3 + a_{z,4}t^4 + a_{z,5}t^5 + a_{z,6}t^6. \tag{A.7}$$

The coefficients $\mathbf{a}$ are found using the following equation (only x is shown, the method is the same for y and z):

$$\mathbf{p}_x = \mathbf{T}\mathbf{a}_x, \tag{A.8}$$

with

$$\mathbf{T} = \begin{bmatrix} x(0) \\ \dot{x}(0) \\ \ddot{x}(0) \\ x(0.5) \\ x(1) \\ \dot{x}(1) \\ \ddot{x}(1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0.5 & 0.5^2 & 0.5^3 & 0.5^4 & 0.5^5 & 0.5^6 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 2 & 6 & 12 & 20 & 30 \end{bmatrix}$$

$$\mathbf{p}_x = \begin{pmatrix} x(0) & \dot{x}(0) & \ddot{x}(0) & x(0.5) & x(1) & \dot{x}(1) & \ddot{x}(1) \end{pmatrix}^T = \begin{pmatrix} p_{0,x} & 0 & 0 & p_{1,x} & p_{2,x} & 0 & 0 \end{pmatrix}^T$$

which can be solved with $\mathbf{a}_x = \mathbf{T}^{-1}\mathbf{p}_x$ since $\mathbf{T}$ is square. Here, $\mathbf{T}$ is a matrix containing the coefficients a on every row for respectively the position, velocity, and acceleration at $t = 0$ s, the position at $t = 0.5$ s, and the position, velocity, and acceleration at $t = 1$ s. The vector $\mathbf{p}_x$ contains the corresponding target values of the polynomial. In this way, the polynomial can be fitted to the position, velocity, and acceleration constraints.

Points $\mathbf{p}_0 = \begin{pmatrix} p_{0,x} & p_{0,y} & p_{0,z} \end{pmatrix}^T$ and $\mathbf{p}_2 = \begin{pmatrix} p_{2,x} & p_{2,y} & p_{2,z} \end{pmatrix}^T$ are respectively the starting ($t = 0$ s) and ending ($t = 1$ s) point of the trajectory, and $\mathbf{p}_1 = \begin{pmatrix} p_{1,x} & p_{1,y} & p_{1,z} \end{pmatrix}^T$ is the point halfway ($t = 0.5$ s), calculated as

$$\mathbf{p}_1 = \begin{pmatrix} \frac{p_{0,x}+p_{2,x}}{2} & \frac{p_{0,y}+p_{2,y}}{2} & \frac{\|p_2-p_0\|}{2l_{\max}}0.25 + 0.05 \end{pmatrix}^T. \tag{A.9}$$

Here, the last factor creates the typical arc of a step, with a minimum height of 0.05 m and a maximum height of 0.3 m, height depending on the length of the step. The length of a step can be $2l_{\max}$ at maximum, since one foot can step from a distance of $l_{\max}$ from one side of the non-stepping foot to a point $l_{\max}$ from the other side of the foot.

The found coefficients $\mathbf{a}$ are then used to update the location of the stepping foot at every controller cycle. To keep in line with the stepping time t_s, the time is scaled before the new location is calculated: $x(t/t_s)$, $y(t/t_s)$, and $z(t/t_s)$.