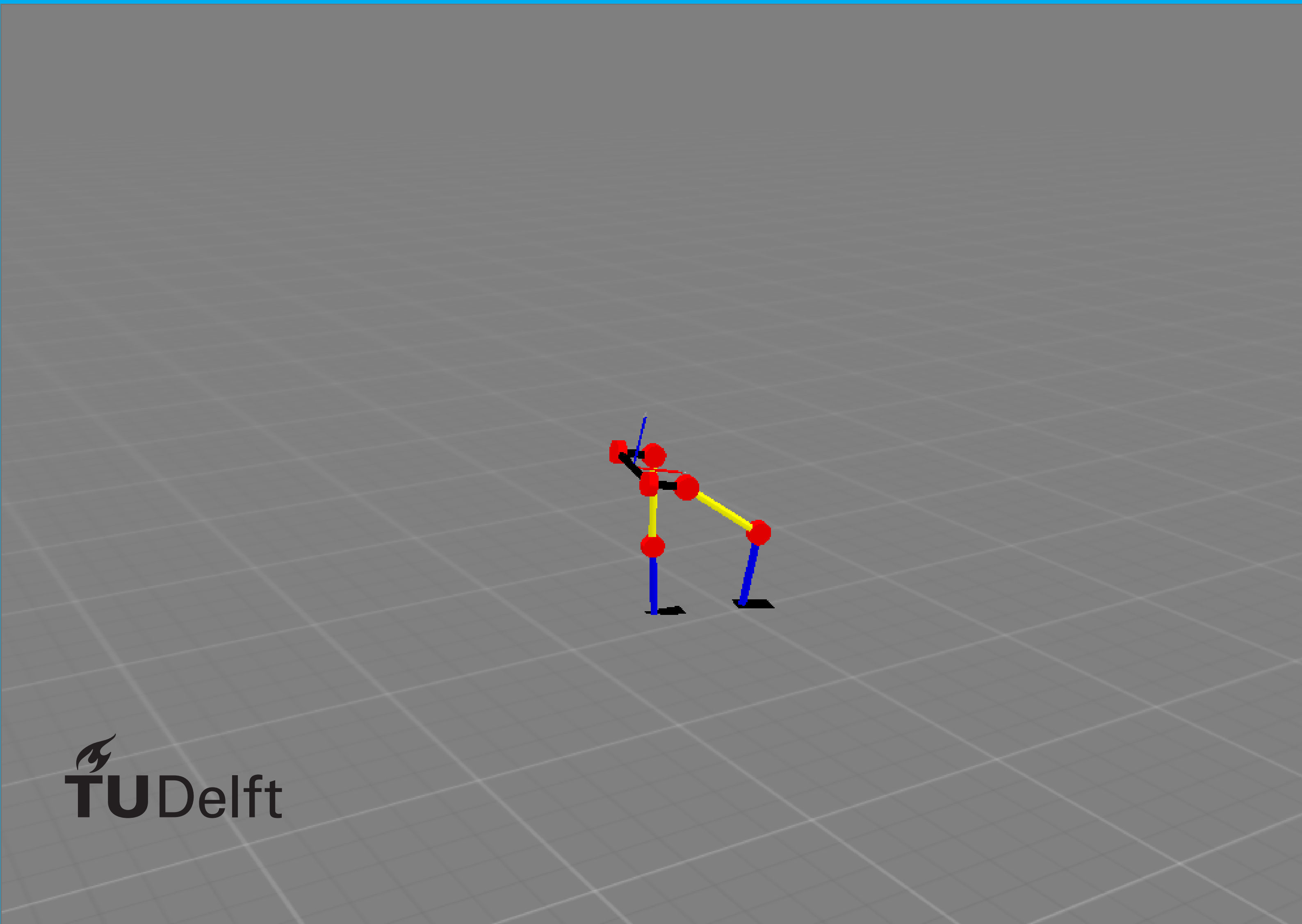
Shared Control in Exoskeletons

A proof-of-concept

O. Siebinga



Shared Control in Exoskeletons

A proof-of-concept

by

O. Siebinga

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday January 15, 2018 at 13:00 PM.

Student number:	4030184	
Project duration:	March 1, 2017 – January 15, 2018	
Thesis committee:	Prof. dr. ir. H. van der Kooij,	TU Delft, supervisor
	Dr. ir. D. A. Abbink,	TU Delft, supervisor

TUDelft

Preface

In October 2016 IHMC robotics' exoskeleton team won the silver medal in the Cybathlon organized by ETH in Zurich. As a team-member I spend several months developing an exoskeleton for this one day race. During the last months the whole team was intensively involved in countless testing and training sessions with our exoskeleton pilot. During these sessions I witnessed the positive effects, both medical and psychological, that this technology had on a team member and friend with a spinal cord injury. But since the ultimate goal of the Cybathlon was to encourage development of new exoskeleton technologies, its tasks also clearly showed the boundaries of state of the art exoskeletons. The limited stability of the exoskeleton proved to be a challenge in all of the Cybathlons tasks.

At the event itself multiple exoskeleton teams from all over the world came together and we discussed how we solved similar problems, exchanged ideas on different strategies and laughed about the big crises we all encountered during development. The general opinion on the stability issue was that it could be solved with full automation, which could be expected from a crowd of robotics engineers. But after these discussions I thought that exploring the possibility of more human control and artificial feedback to the human sounded quite interesting. This master thesis is the result of that interest.

A proof-of-concept of a shared control paradigm for exoskeletons for paraplegics is provided in the journal paper, which forms the main part of this thesis. The paper is supported by some appendices which give a deeper insight in the developed software and the design choices made for the experiment, hardware setup and algorithms. Some raw data and the ethics committee forms are also provided.

O. Siebinga
Delft, December 2017

Contents

1

Scientific Paper

A proof-of-concept of a shared control paradigm for lower body exoskeletons for paraplegics

O. Siebinga BSc,

4030184, BioMechanical Engineering, Faculty 3ME, Delft University of Technology

***Abstract*—Currently, lower-body exoskeletons for paraplegics are investigated as an alternative to wheelchairs and as an exercise method with medical benefits. Literature provides little examples for users to influence the length or frequency of steps taken by the exoskeleton, complicating stability and practical usability. This study proposes a novel control paradigm that allows users to influence step parameters of the exoskeleton, through a bi-directional haptic interface that also provides feedback. The exoskeleton handles the cyclic walking pattern while the patient is enabled to correct for disturbances. I adapted an existing lower-body exoskeleton trajectory generator to allow real-time adaptation of step length and swing time. To demonstrate a proof-of-concept of the control paradigm, I implemented the controller for a virtual 2D exoskeleton, to be controlled by an existing bi-manual haptic control interface. A human-in-the-loop experiment was performed with the goal to compare the benefits of user control over either step length or swing time with a situation with no human control. In the experiment perturbations of increasing magnitude were applied to a 2D virtual exoskeleton, participants could increase or decrease either step length or swing time during swing to correct for these disturbances. The number of successful step taken before the perturbations resulted in a fall were measured. The swing time group succeeded in making the exoskeleton walk stably significantly longer then when there was no human input, proving that the proposed control paradigm is feasible and beneficial for stability.**

I. Introduction

Every year 8 out of every million people suffer from a Spinal Cord Injury (SCI) resulting in paraplegia in Europe and in Northern America there are 19 of these cases per million per year [1]. In the Netherlands and Flanders alone an estimated 12.500 people in total suffer from paraplegia [2]. These patients suffer from a wide range of medical and psychosocial problems [3], [4], [5].

Paraplegic patients having an injury at the T6 level or lower have no control over their lower body, and receive no feedback on its movements or the forces acting on it if they are classified as ASIA-A SCI as defined by the American Spinal Injury Association. However these patients still have full control over their upper body. Most of them use manual wheelchairs in their daily lives. Wheelchairs are an efficient means of transportation for paraplegics but not all environments are wheelchair friendly. Besides that, they require a minimal amount of movement and confine the human to a seated position.

To counter these problems lower limb exoskeletons are used to assist in walking, but only in clinical environments. The development of exoskeletons started in the 1960's and initially focused on augmenting human strength [6]. In later years exoskeletons for rehabilitation and assisted walking were developed. Due to limitations in actuator power-to-weight ratio and computational power the first functional clinical exoskeletons emerged about half a century after the first publications on this subject [7], [8], [9], [10], [11], [12].

Recently much progress has been made in robotic lower limb exoskeletons to assist paraplegics in walking [13], [14], [10], [15], [16]. These exoskeletons offer significant benefits for paraplegic patients like enabling them to ambulate, improving spasticity and bowel movement regularity [7] and even slight recovery in motor function and somatic sensation has been observed in some cases [17]. But currently exoskeletons can only be used in clinical environments under supervision and with balance aids such as crutches or hand rails. If an affordable exoskeleton could be developed that can be used by paraplegics on a daily basis in a home environment the benefits of exoskeletons will become widely available for paraplegics.

The reason that current exoskeletons can only be used under supervision and with balance aids lies in their limited stability, or their ability to cope with disturbances, and thus the high risk of falls. Healthy human beings prevent falling after a disturbance by reacting with foot placement and timing, by adapting the desired joint trajectories real-time (i.e. during a step) [18]. Most exoskeletons are not able to do this because they make use of a state machine where the human triggers the state changes either through direct input or sensors of the device itself based on certain criteria like upper body tilt [14], [10], [15], [19], [16]. Once in the swing state, no adaptions can be made to the calculated trajectories. The only possibility for the human pilot to influence the stability is by controlling his or her upper body, by leaning in a specific direction or by exerting forces to the world . Through the mechanical coupling between the upper and lower body this will influence the stability of the whole system. This is visualized in figure 1.

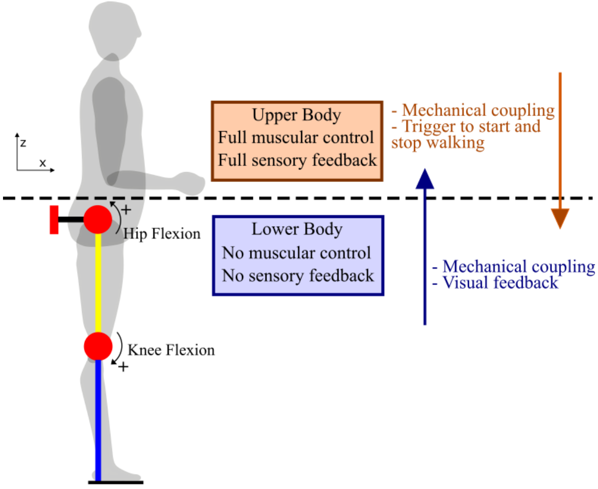


Fig. 1. This figure visualizes the control of a human in a conventional lower body exoskeleton. The upper body and lower body are controlled separately by respectively the human and the exoskeleton. The human has full feedback and full muscular control over the upper body but not over the lower body. The only control the human has over the exoskeleton is a trigger to start and stop walking. The only means of stabilizing comes from the mechanical coupling of the body parts. By exerting forces on the upper body the human can somewhat stabilize his gait. To do this the upper body needs contact with the world, hence the need for crutches or hand rails. The exoskeleton in this figure is in a neutral (0 orientation) position and the positive flexion directions are given by the arrows.

If real-time adaption of the trajectories is to be implemented in exoskeletons it can be done either autonomously by the machine or by incorporating the human in the loop. Currently, the only exoskeleton for paraplegics that does make use of real-time trajectory adaption is the Mindwalker. In 2015 Wang *et al.* succeeded in implementing a Step Width Adaption algorithm hat enabled able-bodied individuals to use the exoskeleton without additional balance support [16]. This is a form of push recovery done autonomously by the machine.

This proves that fully automated adaption is beneficial, but full automation has some pitfalls. The disadvantages of a fully automated system become clear when comparing current exoskeletons to modern humanoid walking machines (e.g. robots like Boston Dynamic's Atlas or NASA's Valkyrie). These robots have far superior sets of sensors with which they do full body state estimation.

Full body state estimation is impossible without accurate measurements of the upper body state which would require a full set of sensors on the upper body. Incorporating this in an exoskeleton would be challenging and costly. Besides that, these robots are fully actuated where an exoskeleton only actuates half the human body. Combined with the fact that the robot controller can determine its own strategy where the exoskeleton always has to cooperate with the human upper body, this can result in potentially dangerous situations if conflicting control strategies are applied.

The opposite strategy would be to leave full control to the human, this would have some advantages. The user is mounted on the exoskeleton, and still has rich visual and vestibular feedback that is beneficial in the task of locomotion, most exoskeletons lack these types of feedback [20], [14], [10]. Combined with artificial feedback from the exoskeleton this may provide valuable contributions to stability. Besides that the sheer computational power and adaptive capabilities of the human being should be considered. A human is aware of his or her surroundings and is capable of correcting the control strategy for it, where machines are bad at handling new situations.

So from a pure control perspective it would make sense to leave complete control over the exoskeleton to the human by controlling all joints independently. But since paraplegics lack all proprioceptive feedback below their injury they are dependent on visual feedback when it comes to determining the orientation of the legs, which is important when estimating the position of their center of mass. In order to decrease the visual load and make complete control possible, extra feedback on the orientation of the legs should be given to the human in a non-visual way. This would require a very complex bi-directional and multi-channel human machine interface. Learning and executing a non natural way of walking with such an interface might be exhausting and success is not guaranteed.

A compromise and a more achievable strategy would be to share the control between the user and the exoskeleton. The EU funded BALANCE project [21] aims to create a controller that uses this strategy for a cooperative exoskeleton, but such an exoskeleton requires human movement control of the lower body and is therefore not suitable for paraplegics who are fully paralyzed below the waist.

To develop a shared control paradigm that is suitable for paraplegics this study proposes to let the exoskeleton plan the regular walking and let the human handle disturbances by altering a simple set of high level control parameters. This could be classified as an adaptable control strategy instead of an adaptive controller, as defined by Oppermann in 1994 [22].

This study aims to provide a proof-of-concept of such a control paradigm, using an existing non-portable haptic interface. Because this interface was non-portable it could not be used on a real exoskeleton, so an experiment was performed in a 2 dimensional computer simulation instead. Since the complexity of the controlled parameters and the haptic interface should be kept minimal a 2D simulation was used instead of a 3D simulation. An existing joint trajectory generation algorithm was adapted so that is capable of handling real-time adaptions to the gait, by altering the parameters of a step (e.g. step length and swing time). An experiment was performed where human test subjects adapted the step parameters real-time and corrected for disturbances. A haptic interface provided feedback on the hip flexion angles. Half of the participants were given control over swing time while the other half were given control over step length. The results of these test subjects were compared to the results of a walking exoskeleton in a stable cycle, without any additional balance control. It was expected that both human groups would perform significantly better then the control group.

II. Control Scheme and Exoskeleton Simulation Design

With this control paradigm the human should be able to influence the gait with a simple set of parameters in order to minimize the complexity of the human machine interface. If these parameters could be chosen such that they match the parameters used to define the gait in existing exoskeleton trajectory generators, then the proposed control scheme could be implemented rather easily. Another benefit would be that the feasibility of that trajectory generator in a real exoskeleton would already have been proven. In this study a trajectory generation strategy was used that was used by IHMC Robotics in their Mina V2 exoskeleton [20]. This trajectory generation strategy uses step length and swing time to define the gait. It is further explained in section II-C.

The haptic interface that was used consists of two handles that can be operated by the hands of the human controller and is further explained in section II-D. The interface gives mechanical feedback on the hip flexion angle to the arms. And controls an exoskeleton computer simulation which is explained in section II-A.

The interaction between the human and the exoskeleton is visualized in figure 2. Because the interaction with this simulation differs from the interaction with a real life exoskeleton the figure contains orange parts, which are disabled in the simulation environment, and blue parts which are present in the experiment.

A. Computer Simulation

The computer simulation was developed in the open source simulation environment ”Simulation Construction Set” which is developed by IHMC Robotics and is available from their website [23]. The parameters used for the exoskeleton are provided in table I. The simulation was set up in a three dimensional environment and the exoskeleton model has six actuated degrees of freedom: hip flexion/extension, hip adduction/abduction and knee flexion/extension. The ankles are modeled as passive springs (for parameters see table I). However in this study only a two dimensional problem was considered, so the hip adduction/abduction actuators were given a 0 reference with high stiffness gain (see table III).

Sideways motions where prevented by a virtual linear spring damper ($k = 1500\frac{N}{m}, c = 500\frac{N}{ms^{-1}}$) perpendicular to the sagittal plane, acting on the pelvis. A virtual rotational spring ($k = 300\frac{Nm}{rad}$) around the z-axis also acting on the pelvis prevented the exoskeleton from turning, so the exoskeleton walks in a straight line. A screen shot of the simulation can be found in figure 3.

At the start of the simulation the exoskeleton assumes a normal squared standing position. This gives the simulation a natural look but causes a few unstable steps at the beginning. So at the first two steps a virtual balance aid prevents the exoskeleton from falling. This aid acts as a virtual spring damper system on the pelvis ($k = 180\frac{N}{m}, c = 200\frac{N}{ms^{-1}}$) perpendicular to the coronal plane and prevents the exoskeleton from falling forwards or backwards. After the first two steps this aid is removed and from this point on the exoskeleton can fall.

Using heuristic tuning the default step parameters, reported in table II, were determined. With these parameters the exoskeleton can walk in a stable cylcle when the balance aid is removed without human input or any automatic balance control. However in the experiment an increasing perturbation force is exerted

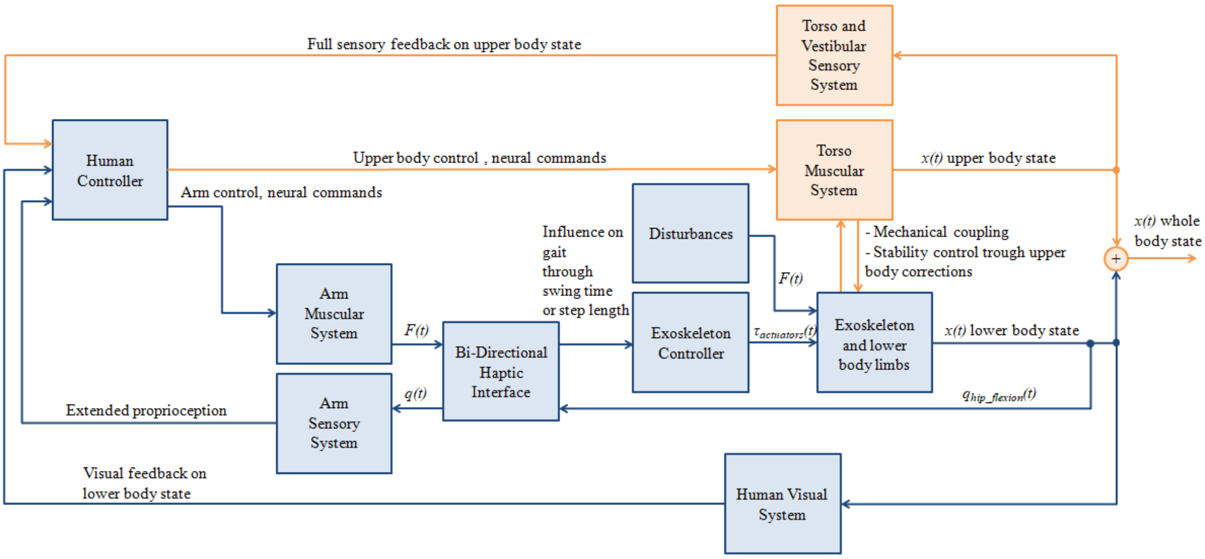

Full sensory feedback on upper body state
Torso and Vestibular Sensory System
Human Controller
Upper body control , neural commands
Torso Muscular System
x(t) upper body state
Arm control, neural commands
Influence on gait through swing time or step length
Disturbances
- Mechanical coupling
- Stability control trough upper body corrections
x(t) whole body state
+
Arm Muscular System
F(t)
F(t)
Bi-Directional Haptic Interface
Exoskeleton Controller
τactuators(t)
Exoskeleton and lower body limbs
x(t) lower body state
Extended proprioception
Arm Sensory System
q(t)
qhip_flexion(t)
Visual feedback on lower body state
Human Visual System

Fig. 2. The control scheme as proposed in this study. In this control scheme the human has influence over the step parameters through a bi-directional haptic interface. In practice this is done by relating forces from the hands to step parameters. Also, artificial proprioceptive feedback is added by feeding back the hip flexion angle to the hands using the same bi-directional haptic interface. Like in current exoskeletons the human is able to influence the lower body state through the mechanical coupling between the upper and lower body. In the simulation however, this coupling does not exist, so the blue parts in this block scheme will be present in the simulation experiment but the orange parts are only present in a real-life exoskeleton.

TABLE I

Parameters of the exoskeleton model used in simulation. The masses are based on other exoskeletons in literature [20] and the dimensions are based on the 50 percentile male. Rotational inertia's of the upper and lower legs were assumed zero for simplicity. All centers off mass are positioned at the center of the link.

Hip Width	$45cm$
Hip Length	$20cm$
Hip I_{xx}, I_{yy}	$15kg\ m^2$
Hip I_{zz}	$10kg\ m^2$
Upper Leg Length	$50cm$
Lower Leg Length	$40cm$
Foot Width	$10cm$
Foot Length	$20cm$
Hip Mass	$12kg$
Upper Leg Mass	$5kg$
Lower Leg Mass	$5kg$
Foot Mass	$2kg$
Ankle Stiffnes	$200\frac{Nm}{rad}$
Ankle Damping	$150\frac{Nm}{rad/s}$
Knee Damping	$2.5\frac{Nm}{rad/s}$
Hip Damping	$2.5\frac{Nm}{rad/s}$

on the pelvis every few steps which will make the exoskeleton fall eventually. This force lasts for $0.2s$ and starts $0.3s$ after toe off. The direction of the force is determined at random at the beginning of each run and is either forwards or backwards. The magnitudes of the

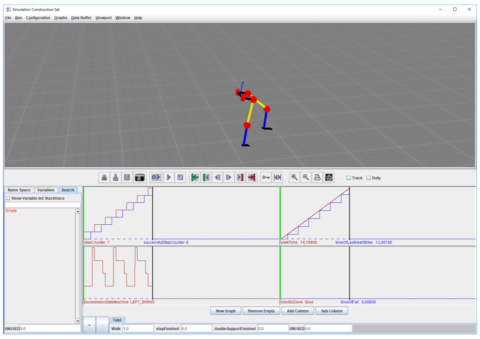


Fig. 3. A screen shot of the simulation of a walking exoskeleton. The red disks represent the actuators, the black bars are part of the hip tube, the yellow bars are the upper legs and the blue bars are the lower legs. The variable plots in the lower part of the window are for evaluation purposes only and were not visible during the experiment.

disturbance forces are also generated at the beginning of each run. For the timing of every next disturbance force, a step interval is picked from a uniformly distributed collection $U(2,4)$. The next disturbance force will occur after this number of steps. The next disturbance force is calculated as $F_n = F^d_{n-1} + x^d$ where $F_1^{forward} = 83\ N$, $F_1^{backward} = -94\ N$ and $x^{forward}$ and $x^{backward}$ respectively are samples from the normally distributed collections $N(5,1)$ and $N(-3,1)$.

TABLE II
DEFAULT STEP PARAMETERS USED IN SIMULATION, WITH THESE PARAMETERS THE EXOSKELETON CAN WALK IN A STABLE CYCLE WITHOUT HUMAN INPUT OR ANY AUTOMATIC BALANCE CONTROL

Swing Time	$1.7s$
Double Support Time	$0.5s$
Step Length	$45cm$
Step Height	$20cm$

B. Exoskeleton Controller

The exoskeleton controller is build around a state machine, a visualization of this state machine can be found in figure 4. It consists of 6 states, the exoskeleton starts in the square state. In this state the exoskeleton is standing square. The state machine transitions to the left swing state when the `Walk` Boolean is true. From this point on the state machine will cycle trough the left and right swing and trailing states based on timing only. So the state transitions when the double support time or swing time is passed. If the hips of the exoskeleton hit the ground during one of the four walking states, the state machine transitions to the isDown state which disables all joint movements.

When one of the swing phases is entered the trajectory generator will initialize, this is explained in the next subsection. In the swing phases the trajectory generator generates joint angle and velocity references for every point in time during swing. During the double support phases, the initial angles of all joints are maintained with a zero velocity reference. The position references are filtered with a 100 Hz low pass filter to prevent peaks in the signal due to calculation errors.

The position and velocity references are then send to the virtual actuator which uses a pd-controller to determine the torque that should be delivered by the actuator. This torque is set in the simulation, in a real life exoskeleton this has to be converted to motor power by a motor amplifier. The gains used by the different virtual actuators can be found in table III.

TABLE III
GAINS OF THE VIRTUAL ACTUATORS. THE HIP ABDUCTION ACTUATOR IS ONLY USED WITH A 0 REFERENCE.

	K_p $[\frac{N}{rad}]$	K_d $[\frac{N}{rad\ s^{-1}}]$
Knee Flexion	600	55
Hip Flexion	1750	100
Hip Abduction	3500	400

C. Trajectory Design

The trajectory generation algorithm used in this study was based on the controller used by IHMC robotics [20] in the 2016 Cybathlon. It uses two trajectory generators, one for the swing leg and one for the stance leg, that generate trajectories every time one of the two swing states is entered. In order to let it handle real-time adaptions to the trajectories a re-initialization procedure was added. A short description of the IHMC algorithm will be provided here followed by the newly implemented re-initialization procedure.

When a step is initialized the upcoming stance foot is used as the reference frame. First the initial and final stance leg hip flexion angles for the upcoming step are determined. The initial angle follows from the current state and the final angle follows directly from the step length, assuming that the final pose is symmetrical. These angles with 0 velocity at time points $t = 0$ and $t = t_{swing}$, where t_{swing} is the swing time, serve as boundary conditions for a minimum-jerk trajectory for the hip flexion joint. The knee of the stance leg is given a zero angle and velocity reference during the whole swing phase.

The next step is to determine the swing leg trajectory, this is done using four way points through which the ankle will pass. The final time of the swing leg trajectory is set at 82% of t_{swing} to make sure the swing leg is not moving at heel strike. The placement of the way points is illustrated in figure 6. The first and the last way point are the initial and final position of the swing foots ankle and are derived in the same way as with the stance leg, based on step length. The intermediate way points are placed at 20% and 80% of the distance between the initial and final way points. Their height is directly determined by the step height variable. The desired velocity vectors at these intermediate points are determined by scaling the distance between the initial or final position and the way point with $\frac{1}{(0.2*0.82)t_{swing}}$.

With the positions and the velocity vectors at these four way points the swing legs joint angles and velocities are determined trough inverse kinematics. For both joints a minimum-jerk trajectory can now be determined. These minimum-jerk trajectories serve as position inputs to the simulated exoskeleton. A flowchart of the initialization procedures can be found in figure 5.

In case either the step length or swing time is altered during the swing phase of a step a re-initialization takes place, this was newly implemented for this study. The re-initialization procedure also starts with the stance leg trajectories. The initial hip angle remains the same, the current angle and velocity are added as an extra boundary condition and the final hip angle is recalculated with the new step length or placed at the new t_{swing}.

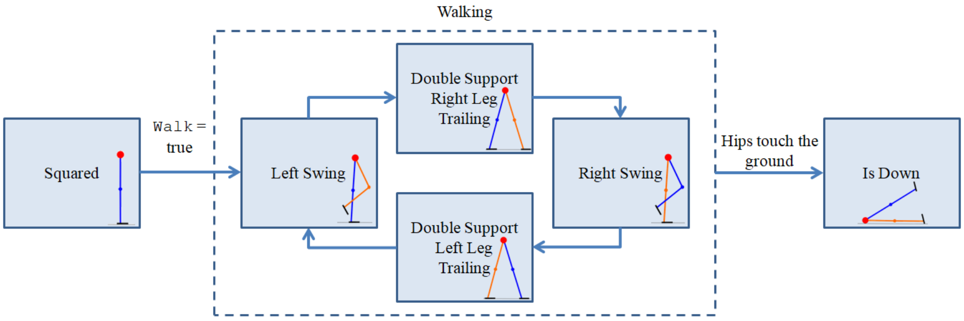


Fig. 4. This block scheme visualizes the state machine of the exoskeleton controller. Every state is represented by a blue block, each block has an illustration to represent the state. In these illustrations the left leg is orange and the right leg is blue. The exoskeleton starts in the squared state in which it assumes a normal standing pose with both feet aligned, when the `Walk` Boolean is true the state machine transitions to the walking phase (represented by the dashed square). All transitions in the walking phase are triggered based on timing. So the state transitions if the swing time or double support time has passed. The human controller has no direct influence on this state machine or the transitions, he or she can only influence the trajectory generation inside the swing states. This state machine is based on the exoskeleton controller used by IHMC robotics, published by Griffin *et al.* [20]

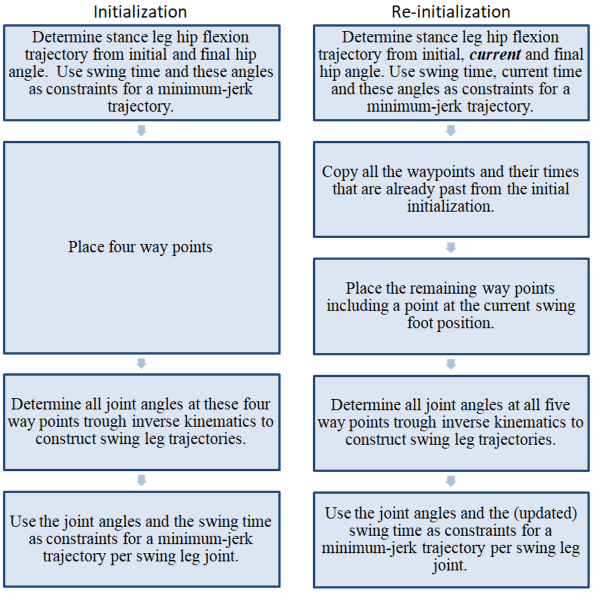

Initialization
Determine stance leg hip flexion trajectory from initial and final hip angle. Use swing time and these angles as constraints for a minimum-jerk trajectory.
Place four way points
Determine all joint angles at these four way points trough inverse kinematics to construct swing leg trajectories.
Use the joint angles and the swing time as constraints for a minimum-jerk trajectory per swing leg joint.
Re-initialization
Determine stance leg hip flexion trajectory from initial, current and final hip angle. Use swing time, current time and these angles as constraints for a minimum-jerk trajectory.
Copy all the waypoints and their times that are already past from the initial initialization.
Place the remaining way points including a point at the current swing foot position.
Determine all joint angles at all five way points trough inverse kinematics to construct swing leg trajectories.
Use the joint angles and the (updated) swing time as constraints for a minimum-jerk trajectory per swing leg joint.

Fig. 5. This flowchart represents the initialization an re-initialization procedures of the trajectory generators. The initialization is executed every time when one of the swing states in figure 4 is entered. The re-initialization is executed when the current step length or swing time differs from the step length or swing time at the last controller tick.

With these 3 boundary conditions a new minimum-jerk trajectory is generated for the stance leg hip flexion actuator.

Then the swing leg trajectories are determined with five way points. Any way points that are already passed are copied from the initial initialization. The current position and velocity of the swing foot are added as a way point and the way points that are still to come are re-calculated with the new step length or placed at a new time point based on the new t_{swing}. This process is illustrated in figure 7. These new way points result in boundary conditions for new minimum-jerk joint trajectories through inverse kinematics. A flowchart of the initialization procedures can be found in figure 5.

D. *Haptic interface*

The haptic interface used in this study was developed by the Delft Haptics Lab as a bi-manual haptic control setup for maritime environments [24], see figure 8. It consists of two actuated handles each mounted on an actuated rotational platform. In this study the rotational platforms were kept in a fixed position by means of a high gain PD-controller ($k_p = 25\frac{Nm}{rad}, k_d = 0.4\frac{Nm}{rad\ s^{-1}}$), leaving only the handles free to move forwards or backwards.

The handles and the rotational platforms are actuated by Maxon motors (part#: 268216) through a cable transmission. The motors are controlled by Maxon motor amplifiers (ESCON 70/10) and they get a torque reference from a real-time Bachmann controller. This Bachmann controller receives position and velocity commands for the handles through an TCP-connection from the exoskeleton simulation. The forces are measured with strain gauges in a Wheatstone bridge. The excitation voltage is provided by a Scaime strain gauge conditioner which also amplifies the measured signal before it is sampled by the Bachmann controller. The force measurements are sent to the exoskeleton simulation over the TCP-connection

The position of the handles was linked to the flexion angle of the hips of the exoskeleton. This way,

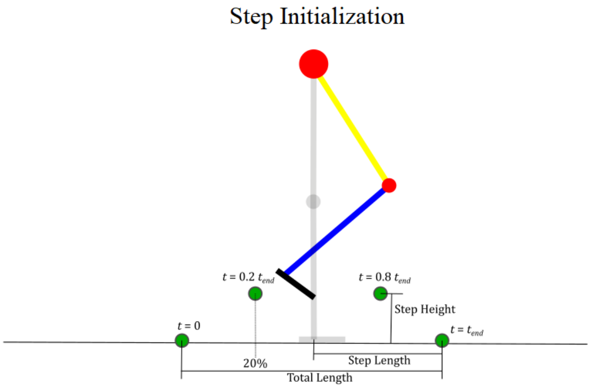


Fig. 6. This illustration shows the initialization of a step, four way points (green dots) for the swing foot are placed. Their positions are illustrated in this figure as green dots with their corresponding time constraints (where $t_{end} = 0.82 * t_{swing}$). These way points result in joint angles and velocities through inverse kinematics which in turn are the boundary conditions for minimum-jerk trajectories for every joint. The swing leg is blue red and yellow, the stance leg is black. The placement of these way points is based on the exoskeleton controller used by IHMC robotics, published by Griffin *et al.* [20]

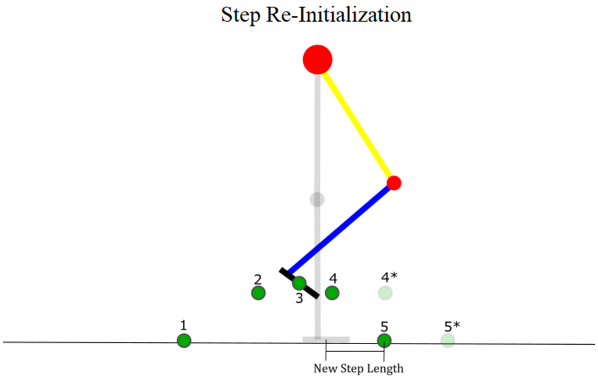


Fig. 7. If one of the step parameters is changed during a step a re-initialization takes place immediately. In this example the step length is altered half way. The way points that were already passed remain the same, these are green dots 1 and 2. A new way point at the current foot position and velocity is added, dot 3. The remaining way points from the initial initialization are discarded (4* and 5*) and replaced by new way points that are calculated with the new step length (3 and 4). In case the swing time is altered, their position remains the same but they are given a new point in time as a boundary condition. These way points result in joint angles and velocities through inverse kinematics which in turn are the boundary conditions for minimum-jerk trajectories for every joint. In this picture the swing leg is shown in blue, red and yellow, the stance leg is black. The used way points are dark green and the discarded way points are light green

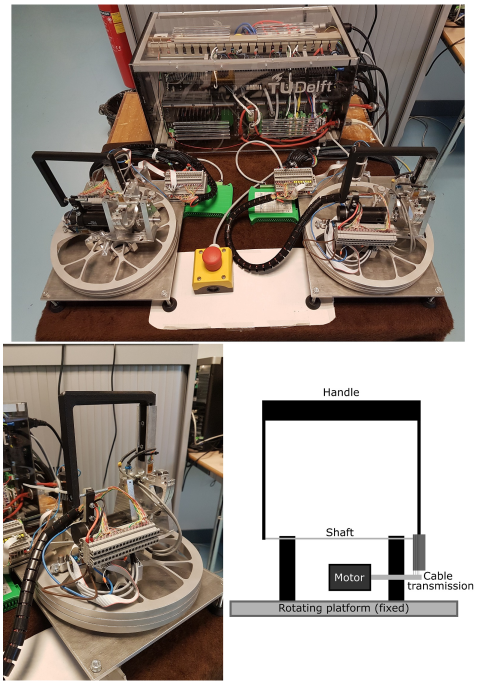


Fig. 8. The control setup for maritime applications, built by the Delft Haptics Lab. The top picture gives an overview of the complete setup. The picture on the bottom left shows a close-up of one handle. And on the right a front view sketch of one handle is shown.

the human controller has an idea on where its legs are without the need of visual attention. The force between the handle and the axle (i.e. the force the human exerts on the device) is measured and used as an input. The inputs and outputs of the setup are visualized in a block scheme in figure 9.

The maximum force exerted by the motors and thus the maximum force that will be measured is $10N$, assuming negligible accelerations. A threshold of $5\ N$ is used to neglect any effects of the inertia of the hands and the handles. The force from both levers is summed, scaled and used as the rate of change for the combined input factor (f_n in figure 9). The increase of the combined input factor per controller step $(0.001s)$ can be calculated as $\frac{F_L+F_R}{50000}$ (see figure 9). In practice the participants tended to use only one of the two handles at the time. This results in a maximum rate of change of the controlled variable of $\frac{10}{50000} * 1000 = 0.2$ per second, which corresponds to a rate of change of $20\% s^{-1}$.

When a swing state is entered the combined input factor is reset to 1.0 and from that point on it is multiplied with the initial swing time or step length as reported in table II. Changes to the steps are only accepted if the current time $t < 75\%\ t_{swing}$. If the new step length is shorter then the current position of the swing foot, the new parameters are also rejected. Furthermore the minimum acceptable swing time is $0.5s$ and the maximum acceptable step length is $1m$. A

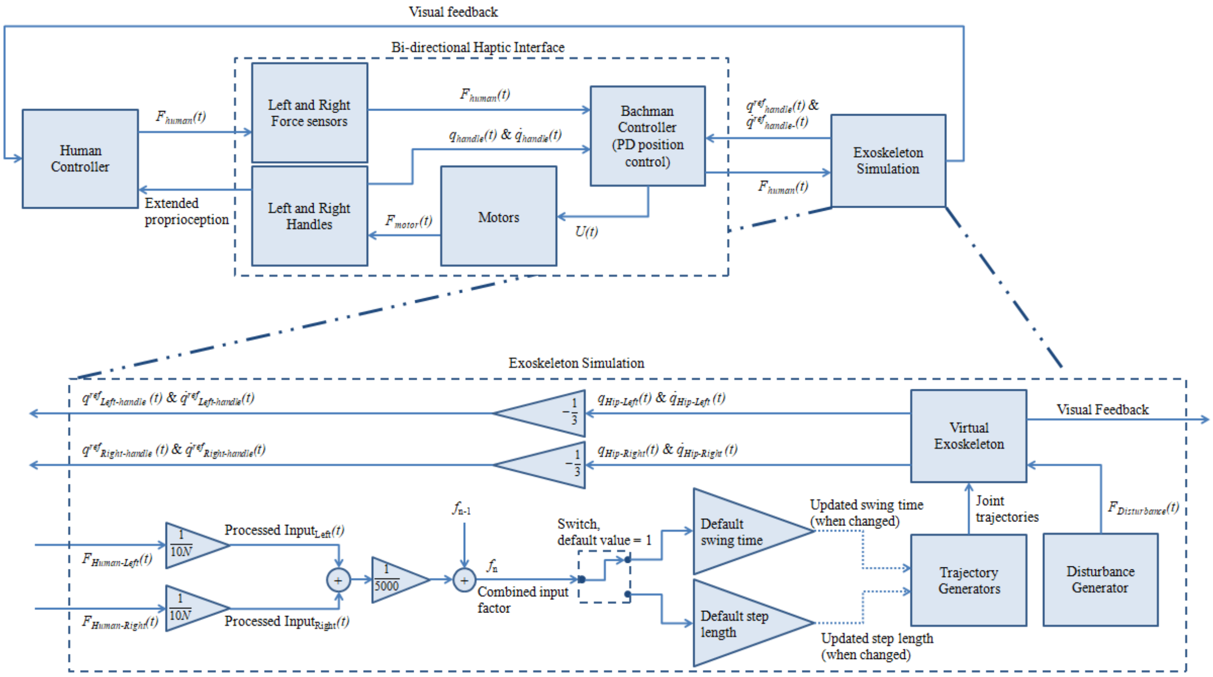

Visual feedback
Bi-directional Haptic Interface
Human Controller
$F_{human}(t)$
Left and Right Force sensors
$F_{human}(t)$
Bachman Controller (PD position control)
$q^{ref}_{handle}(t)$ & $\dot{q}^{ref}_{handle}(t)$
Exoskeleton Simulation
$q_{handle}(t)$ & $\dot{q}_{handle}(t)$
$F_{human}(t)$
Extended proprioception
Left and Right Handles
$F_{motor}(t)$
Motors
$U(t)$
Exoskeleton Simulation
$q^{ref}_{Left\text{-}handle}(t)$ & $\dot{q}^{ref}_{Left\text{-}handle}(t)$
$-\frac{1}{3}$
$q_{Hip\text{-}Left}(t)$ & $\dot{q}_{Hip\text{-}Left}(t)$
Virtual Exoskeleton
Visual Feedback
$q^{ref}_{Right\text{-}handle}(t)$ & $\dot{q}^{ref}_{Right\text{-}handle}(t)$
$-\frac{1}{3}$
$q_{Hip\text{-}Right}(t)$ & $\dot{q}_{Hip\text{-}Right}(t)$
$f_{n\text{-}1}$
Switch, default value = 1
Default swing time
Updated swing time (when changed)
Joint trajectories
$F_{Disturbance}(t)$
$\frac{1}{10N}$
Processed Input$_{Left}(t)$
$F_{Human\text{-}Left}(t)$
+
$\frac{1}{5000}$
+
f_n
Combined input factor
Trajectory Generators
Disturbance Generator
$\frac{1}{10N}$
$F_{Human\text{-}Right}(t)$
Processed Input$_{Right}(t)$
Default step length
Updated step length (when changed)

Fig. 9. This block scheme shows the interaction between the human controller, the bi-directional haptic interface and the exoskeleton simulation, the lower dashed square illustrates the processes inside the exoskeleton simulation. The bi-directional haptic interface has an internal PD-controller which uses the reference angles that follow from the simulation to control the handle orientation. This serves as extended proprioception to the human controller. The human controller exerts forces on the handles which are measured with force sensors, sampled by the Bachmann controller and send to the simulation. Here, the forces are scaled to a unit-less processed input between -1 and 1. These processed inputs are scaled and added to the previous value of the combined input factor resulting in a new combined input factor f_n. This factor is reset to 1 every time when a swing state is entered (see figure 4). The combined input factor is multiplied with either the default swing time or the default step length, depending on the experimental conditions. If a change occurs in either the swing time or step length, the updated variable is send to the trajectory generators where it triggers a re-initialization (see figure 7).

practical example to illustrate this working principal is shown in figure 10.

III. Experimental Methods

A. Subjects

Twelve subjects (10 male, 2 female, ages 18-29, mean age 23.33) volunteered to participate in the experiment. They had no experience controlling exoskeletons nor any pre-existing knowledge on the experiment or the controller. All subjects gave their written informed consent prior to the experiment. The setup and experiment were approved by the ethics committee of the Delft University of Technology.

B. Procedure

The participants were asked to make the exoskeleton walk for as long as possible, a run lasted until the exoskeleton fell down. The subjects were divided in two groups, 6 of the subjects where given control over the swing time while the other 6 were given control over the step length. The subjects were instructed to sit behind the handles and hold them with a light grip, to minimize the measuring of inertia effects. The subjects were instructed not to overrule the position of the handle, because overruling the position would not increase the input since the measured force in that case is always equal to the maximum motor force. Besides that, by forcing the handle in another position all position feedback is lost.

After this instruction the subjects were shown what the simulation looks like and how far the exoskeleton can walk without any human input. During this run without human input a short explanation on the disturbance generator was provided. The subjects were explained that the exoskeleton would be pushed every few steps. They were also told that the magnitude of the pushes increased over time and that all the pushes during one run would be in the same direction. All participants received the hint that carefully looking at the velocity of

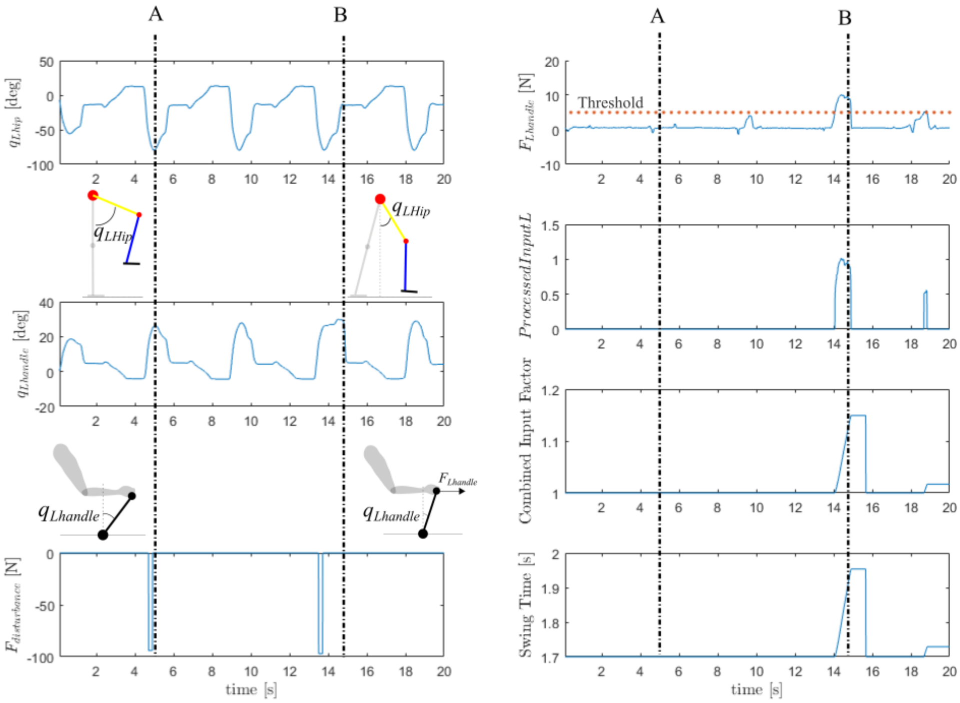


Fig. 10. This figure shows the first 20 seconds of a run with swing time control to illustrate the working principal of the feedback and input, it only shows the angles and forces on the left leg and the left handle for simplicity. At point A the left (swing) leg angle is at a minimum as can be seen in the top left plot, this corresponds with the illustration next to the dotted line marked A. This leg angle results in a positive angle for the left handle as is shown in the middle left plot and in the lower illustration next to the dotted line A. The lower left plot shows that at point A a disturbance just occurred. However the human controller did not respond to this disturbance as can be seen in the top right plot. Since the force on the handle remains under the threshold value (top right plot) for the first 14 seconds, the default value for swing time ($1.7\ sec$) is not changed during this period. The exoskeleton walks without adapting the gait for the disturbance. At point B however, just after the second disturbance the human controller exerted a positive force (as defined in the lower illustration at line B) greater then the threshold. This results in a positive processed input which is a scaled version (range $-1, 1$) of the input force. This in turn results in an increase of the combined input factor which stops as soon as the input force gets below the threshold. The combined input factor is constantly multiplied with the default swing time, resulting in an increase. This triggers a re-initialization of the trajectory generators which causes the cycle directly after B to be slightly longer then the first cycles. When step length control is used, the threshold and factors are the same but the combined input factor is multiplied with the default step length instead of with the default swing time.

the pelvis might help them in estimating when and how to react.

After this demonstration the subjects were told which parameter they would be controlling. The fact that they should only correct if they thought the exoskeleton was unstable was explicitly mentioned.

Every subject completed 30 runs in total. After every 5 runs there was a short break of approximately 2 minutes and after the first 15 runs there was a 15 minute break. After every run the participants were given their last score, they were told what the high-score of their group was at the beginning of the experiment.

The first set of five runs was a familiarization phase, during these runs real time plots of the combined input factor and the swing time or step length were visible to give the participants insight in the magnitude of their corrections. Additionally plots of the position error of the handles were shown to point out if the subject was overruling the handles position. The second set of five runs was the training phase, during these two phases all instructions were repeated if necessary. Runs 11-30 were recorded as data points, after the 15 minute break one extra training-run was executed before the set of five to make sure the subject was focused. A flow chart of the procedure is given in figure 11.

C. *Control strategy*

In order to stabilize the exoskeleton the swing time group had to shorten the swing time when the exoskeleton was falling forwards and lengthen it when the exoskeleton was falling backwards. The step length groups had to lengthen the steps when the exoskeleton

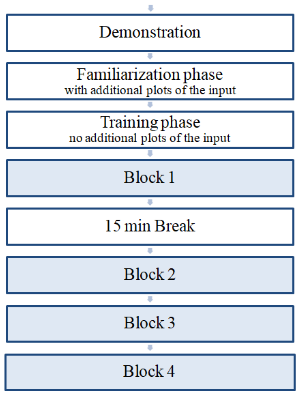


Fig. 11. A flow chart representing the experimental procedure executed by each of the 12 participants, of which 6 were in the step length group and 6 in the swing time group. All blocks consist of 5 runs except the demonstration, which consists of only 1 run. Between the blocks there was a 2 minute break every time. The blue blocks represent the blocks in which data was collected.

was falling forwards and shorten the steps when the exoskeleton was falling backwards.

D. Measured Signals

The simulation environment allows for the whole run to be saved as binary data, this has the benefit that the complete run including visualization can be opened and played back. The signals that were used in the data analysis are the human input, recorded as the combined input factor, the disturbance signal and the number of successfully executed steps before the fall. A step was counted as successful if the swing foot touched the ground in the last 18% of swing time under the condition that the exoskeletons pelvis did not touch the ground first.

The human input and disturbance signal were used to analyze the human control strategy and to determine if the fall was due to human error. The number of successfully completed steps is a metric for stability.

E. Data Analysis

To analyze the number of falls due to human error the number of occasions were the exoskeleton fell in the opposite direction of the disturbance force were counted. In order to measure the accuracy of the human corrections all occasions where the human corrected in both directions during one run were counted. Because this indicates that the human over-corrected and had to correct in the opposite direction afterwards. Since this data is nominal a Pearson Chi-squared test was used to test for significance.

To compare the performance in terms of stability of the two different groups, the number of successfully completed steps of the runs are grouped and compared to the results of 120 runs without human input or additional balance control. A Mann-Withney statistical test was used to test for significance since the data was ordinal and not normally distributed.

To illustrate the control effort of the swing time and step length control a measure for control effort was constructed as $\sum |f_n - 1|$ where f_n is the combined input factor (see figure 9). This measure is not only dependent on the controlled parameter but also on the difficulty which is not measureble and increases with the number of steps. For this reason no statistical analysis was performed, the control effort measure is only used to illustrate the typical human behavior during the experiments.

IV. Results

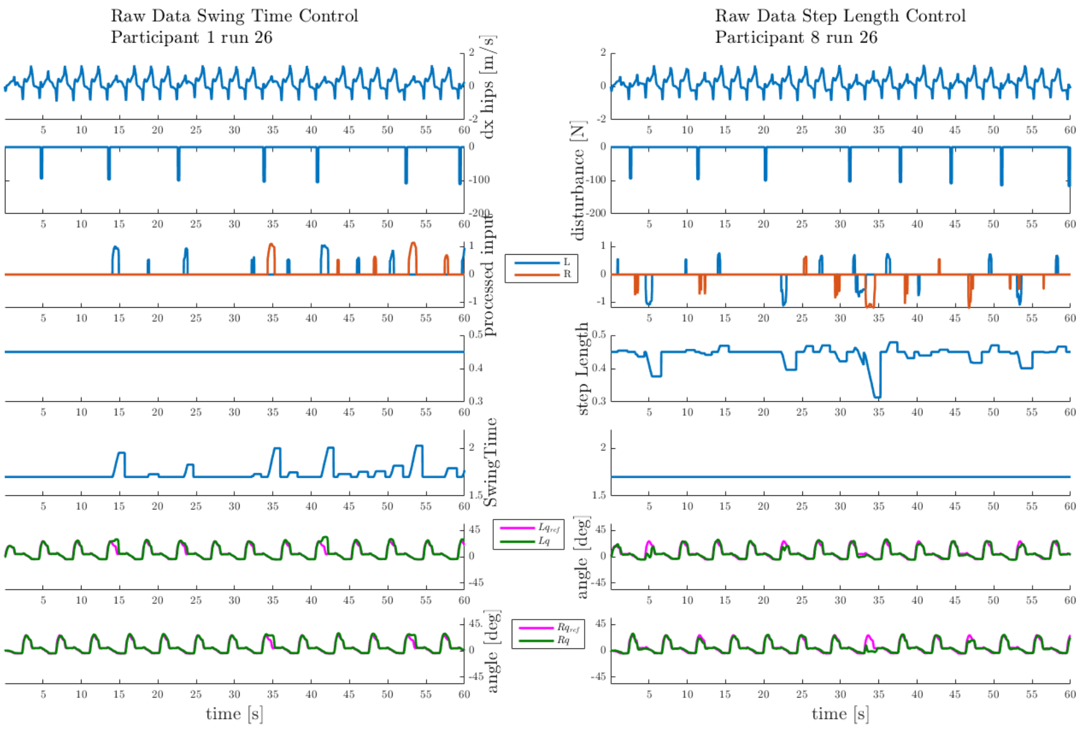
To illustrate the type of results one typical run of a participant from the swing time control group and a typical run from a participant from the step length control group are shown in figure 12.

The results of the groups in terms of stability can be found in figure 13. The statistical test showed that the group with control over swing time performed significantly better (mean rank = 141.87) then when there was no human intervention (mean rank = 99.13) U = 4636.0, p = 0.000. The group with control over step length however performed significantly worse (mean rank = 110.5) then when there was no human intervention (mean rank = 130.5) U = 6000.5, p = 0.024.

The number of falls due to human error the in group that had control over swing time is 7 out of 120. In the group that had control over step length this occurred 39 out of 120 times. But the exoskeleton fell in the same direction as the disturbances every time when there was no human intervention, showing that this is a measure for human induced errors. When comparing the groups they are all significantly different (exo-swing time:$\chi = 7.2, p = 0.007$, exo-step length:$\chi = 46.6, p = 0.00$, step length-swing time:$\chi = 27.5, p = 0.00$,)

In the swing time group there were 11 out of 120 runs with corrections in both directions. In the group with control over the step length this happened 69 out of 120 times. These results are also shown in table IV. The difference between the two groups was significant ($\chi = 63.1, p = 0.00$). A plot of the measure of effort versus the performance can be found in figure 14.

Four out of the six participants of the step length group spontaneously stated that they had difficulties with estimating the amount of correction that


Raw Data Swing Time Control
Participant 1 run 26

Raw Data Step Length Control
Participant 8 run 26

Fig. 12. The first 60 seconds of the data of one run from a participant from the swing time control group and the step length control group. The upper plot shows the forward velocity of the hips. This velocity shows lower peaks at the instances where the second plot shows disturbance forces, which are directed backwards in both runs. The human subject should correct for this velocity loss. The third plot shows the corrections of the subject as processed input (left and right) scaled between -1 and 1. This directly influences the rate of change of the combined input factor (f_n) which is multiplied with the swing time or step length, shown in the fourth and fifth plot. The last two plots show the reference and measured angles of the left and right handles.

TABLE IV
THIS TABLE SHOWS DATA ON THE DIRECTIONS OF THE FALLS AND THE DIRECTION OF THE CORRECTIONS MADE BY THE HUMAN. THE FIRST ROW SHOWS IN HOW MANY OF THE 120 RUNS DONE IN EACH GROUPS THE EXOSKELETON FELL IN THE OPPOSITE DIRECTION OF THE DISTURBANCES. THIS IS A MEASURE FOR THE LEVEL OF FALLS CAUSED BY THE HUMAN CONTROLLER. THE SECOND ROW SHOWS THE NUMBER OF RUNS OUT OF THE 120 RUNS IN EACH GROUP IN WHICH THE HUMAN CONTROLLER MADE CORRECTIONS IN BOTH DIRECTIONS, INDICATING THAT THE HUMAN WAS CORRECTING FOR SELF INDUCED DISTURBANCES. ALL DIFFERENCES BETWEEN GROUPS ARE SIGNIFICANT.

	No Human	Swing Time	Step Length
Falls in opposite direction	0	7	39
Corrections in both directions	-	11	69

was needed. They also stated that over-correction was a problem, to counter this they had to correct in the opposite direction at the next step. Two of these participants also stated that the direction of the correction needed for a fall was counter intuitive since they wanted to pull the handle back when the exoskeleton started falling forwards instead of pushing it forward.

V. DISCUSSION

The two tested control strategies both gave significantly different results from the performance of the exoskeleton without any human intervention. The step length control however performed worse from a stability point of view, the reason for that will be discussed here.

A. Discussion of Results

The reason that the step length group performed significantly worse than the controller without human input is twofold. The first part of this reason lies in the dynamics of the exoskeleton and its controller. The second part can be explained by the statements made by the participants of this group.

To start with the first part, in contrary to regular human walking this exoskeleton controller executes

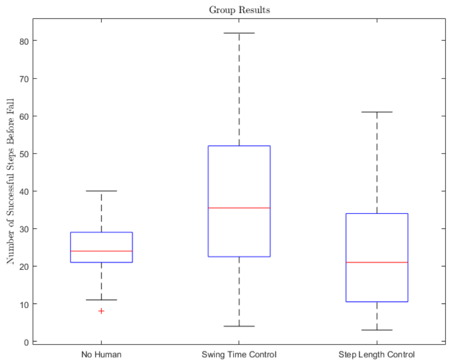


Fig. 13. The results of the two participant groups, 120 samples each, versus 120 data samples of the simulation without any human intervention. The swing time group performed significantly better then the no human group. The step length group performed significantly worse.

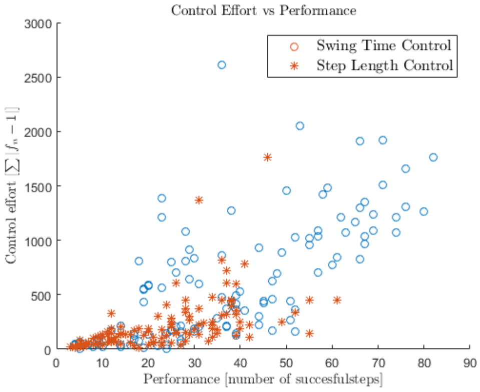


Fig. 14. The control effort of both groups versus the number of successfully completed steps. The control effort is measured as the sum of the absolute value of the combined input factor $f_n - 1$. Since the difficulty is not measurable and increases with the number of successfully completed steps no statistical analysis of this data is performed, this plot only serves as an illustration to the data.

a position based trajectory. The only limit to whether this trajectory is successfully followed is the power in the actuators. If the actuators would have unlimited power the two dimensional and constrained exoskeleton will never fall forwards or backwards. However the actuators controllers are limited in power due to their PD-design, so their acceleration is limited. This explains that giving the actuators more time to accelerate the center of mass helps to stabilize the exoskeleton if it is falling backwards. When the exoskeleton is falling forwards shortening the trajectory time helps stabilizing the exoskeleton for the same reason because the swing time is altered to match the high velocity that cannot be lowered in time. But if the step length is altered, the actuators have little benefits in accelerating or decelerating the center of mass. This partly explains why swing time control is more beneficial then step length control.

The statements of the participants on the other hand indicated two opportunities for improvement in the step length design. The first one being the counter intuitive direction of corrections. If the exoskeleton is falling forwards the swing time participants had to shorten the swing time in order to stabilize the exoskeleton. So they had to exert a force in the opposite direction of the disturbance which apparently was no problem since none of the participants mentioned anything regarding the direction of force. The step length group however had to increase the step length and thus push the lever in the same direction as the disturbance, two of the participants stated that this felt unnatural.

A possible solution to this problem would be to invert the coupling between the direction of the force on the handles and the change in step length. But altering the step length in an exoskeleton seems to have more potential then increasing stability alone. The lack of control over foot placement is a known problem in exoskeleton walking and if this could be fixed exoskeletons would be a little better suited for use in daily live. So inverting the input, might make the control more intuitive from a stability point of view but it would make the system less suitable for controlling foot placement.

It would also be an option to exchange the input and feedback signals, in that case the positions of the handles would be measured as input and the force on the handle would provide feedback. However, position feedback was chosen over force feedback in the current study since the feedback represents an orientation and this way the feedback is modality-matched.

Instead of inverting the controls it might be beneficial to add artificial feedback on the current step length or final foot placement. In a real life exoskeleton, augmented reality offers a lot of potential. If the final placement of the foot could be shown to the human controller this way, it might not only increase the potential for stability but it could also help in real life situations were foot placement is important.

Another option that could be investigated is giving the human control over a combination of step length and swing time. This could combine the stability of the swing time and the practical advantages of foot placement.

The second opportunity for improvement is that step length has a bigger potential to destabilize the exoskeleton, this can be concluded from the greater number of falls in the opposite direction from the disturbances. Since this type off fall only occurred with human control it can be concluded that they are caused by the human interference instead of the disturbance. This could be caused by the fact that there was no indication available on what the current step length was, the participants had no clue how much they were correcting. That conclusion is supported by the fact that the step length group had to correct for their own over-corrections, in the opposite direction, more often. Besides that figure 14 shows that the step length group tended to make less control effort then the swing time group at the same level of difficulty. This could indicate that the step length participants had no understanding of the effects of their corrections and thus were more reluctant to make corrections.

In figure 12 raw data from two individual runs is shown, in these figures it should be noted that in both cases the reference trajectory of the handles is followed accurately with a few exceptions. This means that the subjects didn't violate the position of the handles most of the time. When they did violate the handles position all useful feedback was lost but there was no increase in input. To prevent this from happening a more powerful haptic interface could be used, this would make it more difficult for the human controller to overpower to motors and lose all feedback. It can also be seen that the handle that is used to make the corrections is always the swing leg handle (no exceptions to this observation could be found in the data). These figures also illustrate that the swing time participants only corrected in one direction while the step length participants constantly made contradicting corrections.

B. Future Work

In a real exoskeleton the user has additional cues available, such as upper body proprioception, visual flow, vestibular feedback, and auditory cues from actuators and foot contact. Since these cues were absent in this study it can be expected that the control performance of the human in real life is even better than in this experiment, the benefit of the extended proprioception however might be less in real life. Further research could be done to evaluate the human behaviour in the more demanding real life situation. This is important not only because the added feedback might conflict or increase the difficulty of the task but also because the harmful consequences of falling when the system doesn't work or when the human introduces instabilities might influence the humans behaviour.

Since this study aimed to provide a proof-of-concept of a shared control paradigm and test if this control paradigm allows the human input to increase the stability of an exoskeleton, unrealistic high disturbances were used. More research is needed to quantify the actual stability in a real life exoskeleton. Before that, a similar simulation study could be performed to investigate the behaviour in a three dimensional environment.

In order to perform a real life study custom portable hardware should be developed. Because the eventual stability is unknown and because it is likely that there are instabilities in the sideways direction which demand crutches to counter, it would be sensible to design a form of handles/joysticks that can be integrated in hand rails or crutches. This could be done in the form of thumb-joystick which would leave the rest of the hand free to hold and use the crutches. If the control paradigm has proven to be safe for use without crutches, a stand alone hand held device for control could be developed.

Besides the stability benefits of this control paradigm, an additional benefit compared to the current situation might be that the user experiences a higher sense of agency and influence instead of being walked without the ability to influence the 'walking machine', for paraplegics this could be a great psychological advantage. A study could be performed to verify this.

VI. Conclusion

A human-in-the loop experiment was performed in a 2D computer simulation of a walking exoskeleton to provide a proof-of-concept for a novel control paradigm for lower body exoskeletons for paraplegics. The experiment used an adapted version of an existing trajectory generating algorithm that was able to handle real time adaptions to the gait. In the experiment the exoskeleton handled the cyclic walking motion while human test subjects corrected for disturbances through a bi-manual haptic control interface. The following conclusions can be drawn:

- When given the possibility to adapt the swing time of an exoskeleton gait in real time a human can increase the stability of a walking exoskeleton significantly (mean rank = 141.87 vs 99.13, U = 4636.0, p = 0.000) using only visual feedback and extended proprioceptive feedback on the hip flexion angles.
- When given the possibility to adapt the step length in real time the subjects performed significantly worse (mean rank = 110.5 vs130.5, U = 6000.5, p = 0.024) then the exoskeleton alone using only visual

feedback and extended proprioceptive feedback on the hip flexion angles. So under certain circumstances the risk of a fall increases when a human is incorporated in the loop. The data suggests that the absence of feedback on the magnitude of the correction might be part of the problem in this specific case.

- A proof-of-concept of the shared control paradigm was provided showing that shared control is another feasible option to increase stability in lower body exoskeletons for paraplegics besides full automation of disturbance rejection.

Acknowledgment

I would like to thank the supervisors of this MSc thesis project; prof. dr. ir. H. van der Kooij and dr. Ir. D.A. Abbink for their help and valuable feedback. I would also like to thank dr. ir. H. Boessenkool for his help with the hardware setup and setting up the experiment.

References

[1] B. B. Lee, R. A. Cripps, M. Fitzharris, and P. C. Wing, "The global map for traumatic spinal cord injury epidemiology: update 2011, global incidence rate.," *Spinal cord*, vol. 52, no. 2, pp. 110–6, 2014.

[2] Dwarslaesiefonds, "http://www.dwarslaesiefonds.nl/over-dwarslaesie/feiten-en-cijfers," 2017.

[3] R. Levi, C. Hultling, M. S. NASH, and A. Seiger, "The Stockholm spinal-cord injury study 1. Medical problems in a regional SCI population," *Paraplegia*, vol. 33, no. 6, pp. 308–315, 1995.

[4] R. Levi, C. Hultling, and a. Seiger, "The Stockholm Spinal Cord Injury Study. 3. Health-related issues of the Swedish annual level-of-living survey in SCI subjects and controls.," *Paraplegia*, vol. 33, no. 12, pp. 726–730, 1995.

[5] R. Levi, C. Hultling, and a. Seiger, "The Stockholm Spinal Cord Injury Study: 2. Associations between clinical patient characteristics and post-acute medical problems.," *Paraplegia*, vol. 33, no. 10, pp. 585–594, 1995.

[6] B. R. Fick and J. B. Makinson, "FINAL REPORT ON HARDIMAN I PROTOTYPE FOR MACHINE AUGMENTATION OF HUMAN STRENGTH AND ENDURANCE," tech. rep., 1971.

[7] G. Colombo, J. Matthias, and R. Schreier, "Treadmill training of paraplegic patients using a robotic orthosis," *Journal of Rehabilitation Research and Development*, vol. 37, no. 6, pp. 693–700, 2000.

[8] A. Zoss, H. Kazerooni, and A. Chu, "On the mechanical design of the Berkeley Lower Extremity Exoskeleton (BLEEX)," *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, no. April, pp. 3132–3139, 2005.

[9] J. F. Veneman, R. Kruidhof, E. E. G. Hekman, R. Ekkelenkamp, E. H. F. Van Asseldonk, and H. Van Der Kooij, "Design and evaluation of the LOPES exoskeleton robot for interactive gait rehabilitation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 15, no. 1, pp. 379–386, 2007.

[10] P. D. Neuhaus, J. H. Noorden, T. J. Craig, T. Torres, J. Kirschbaum, and J. E. Pratt, "Design and evaluation of Mina: A robotic orthosis for paraplegics," *IEEE International Conference on Rehabilitation Robotics*, 2011.

[11] E. S. A. Estec, J. Gancet, M. Ilzkovitz, G. Cheron, Y. Ivanenko, H. van der Kooij, F. van der Helm, F. Zanow, and F. Thorsteinsson, "Mindwalker: A brain controlled lower limbs exoskeleton for rehabilitation. Potential applications to space," *11th Symposium on Advanced Space Technologies in Robotics and Automation*, no. April, pp. 12–14, 2011.

[12] R. J. Farris, H. A. Quintero, and M. Goldfarb, "Preliminary evaluation of a powered lower limb orthosis to aid walking in paraplegic individuals," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 19, no. 6, pp. 652–659, 2011.

[13] G. Colombo, M. Joerg, R. Schreier, and V. Dietz, "Treadmill training of paraplegic patients using a robotic orthosis," *Journal of rehabilitation research and development*, vol. 37, no. 6, pp. 693–700, 2000.

[14] A. Esquenazi, M. Talaty, A. Packel, and M. Saulino, "The ReWalk Powered Exoskeleton to Restore Ambulatory Function to Individuals with Thoracic-Level Motor-Complete Spinal Cord Injury," *American Journal of Physical Medicine & Rehabilitation*, vol. 91, no. 11, pp. 911–921, 2012.

[15] H. a. Quintero, R. J. Farris, and M. G. Members, "Control and Implementation of a Powered Lower Limb Orthosis to Aid Walking in Paraplegic Individuals," vol. 19, no. 6, pp. 652–659, 2012.

[16] S. Wang, L. Wang, C. Meijneke, E. V. Asseldonk, T. Hoellinger, G. Cheron, Y. Ivanenko, V. L. Scaleia, F. Sylos-labini, M. Molinari, F. Tamburella, F. Thorsteinsson, M. Ilzkovitz, J. Gancet, Y. Y. Nevatia, R. Hauffe, H. V. D. Kooij, E. van Asseldonk, T. Hoellinger, G. Cheron, Y. Ivanenko, V. La Scaleia, F. Sylos-labini, M. Molinari, F. Tamburella, I. Pisotta, F. Thorsteinsson, M. Ilzkovitz, J. Gancet, Y. Y. Nevatia, R. Hauffe, F. Zanow, H. van der Kooij, E. V. Asseldonk, T. Hoellinger, G. Cheron, Y. Ivanenko, V. L. Scaleia, F. Sylos-labini, M. Molinari, F. Tamburella, F. Thorsteinsson, M. Ilzkovitz, J. Gancet, Y. Y. Nevatia, R. Hauffe, and H. V. D. Kooij, "Design and Control of the Mindwalker Exoskeleton," *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 4320, no. c, pp. 277–86, 2014.

[17] J. Liu, M. Abd-El-Barr, and J. H. Chi, "Long-term Training with a Brain-Machine Interface-Based Gait Protocol Induces Partial Neurological Recovery in Paraplegic Patients," *Neurosurgery*, vol. 79, no. 6, pp. N13–N14, 2016.

[18] M. Vlutters, E. H. F. V. Asseldonk, and H. V. D. Kooij, "Center of mass velocity-based predictions in balance recovery following pelvis perturbations during human walking," pp. 1514–1523, 2016.

[19] K. A. Strausser and H. Kazerooni, "The development and testing of a human machine interface for a mobile medical exoskeleton," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4911–4916, 2011.

[20] R. J. Griffin, T. Cobb, T. Craig, M. Daniel, N. van Dijk, J. Gines, K. Kramer, S. Shah, O. Siebinga, J. Smith, and P. Neuhaus, "Stepping Forward with Exoskeletons: Team IHMC's Design and Approach in the 2016 Cybathalon," *IEEE Robotics & Automation Magazine*, pp. 1–1, 2017.

[21] BALANCE, "http://balance-fp7.eu/," 2017.

[22] R. Oppermann, *Adaptive User Support: Ergonomic Design of Manullay and Atuomatically Adpatable Software*. 1994.

[23] IHMC Robotics, "https://ihmcrobotics.github.io/," 2017.

[24] F. Hoeckx, A. Vrijdag, and D. A. Abbink, "Development of a test setup for exploring the potential of haptic feedback for maritime operations," 2017.

2
Controller

This chapter will provide information on the choices made when designing the exoskeleton controller. It is divided in four subsections. The first subsection will discuss the choice of Java as the programming language and the choice of the open source simulation environment. The second subsection looks into the requirements that were drafted before design and the resulting choices in gait generation and adaption. The third subsection will discuss the ability of the controller to generate and alter three dimensional gaits and why this ability was not used. And the final subsections gives a brief class overview of the complete controller.

2.1. Java and SCS

The exoskeleton controller used in this study was fully written in Java. This was done because in contrary to Matlab for example, Java is a fast executing, object oriented programming language which makes it ideal for robot controllers. The fact that the code is OS independent was also beneficial since the hardware was not known at the start of the project. A Git repository was used for version control and to make the code accessible from multiple machines. The Gradle build tool was used to manage dependencies on third party software.

The simulation environment in which the controller was used is the open source simulation construction set (SCS) developed by IHMC Robotics [2]. This simulation environment was chosen because is is fast enough to provide real-time simulations and has build in visualization. SCS uses a concept called yo-variables, these variables are saved in registers and can be accessed and plotted during simulation on the simulating computer or on a remote visualizer. This is beneficial since it means that the controller can run on a separate real time system while all variables are accessible on an operator computer. Such a setup was eventually not used but as said before, the software was chosen before the hardware so the possibility alone was an advantage.

The yo-variable registers from SCS can also easily be saved and loaded in SCS, this feature was used to store all the collected data. At the start of every run the participants number, age, gender and the trial number were collected and stored in the registry as yo-variables. The participant information was automatically copied from the last run and the trial number was incremented. So the participants information had to be entered only once. At the end of every run all registries were saved in a binary data file. The last state of the robot (i.e. the final values of all variables) were also saved in a separate *.csv file making them easily accessible in Matlab.

2.2. Gait Generation

Different forms of gait generation have been used in exoskeletons, a suitable existing algorithm was searched for because that would already have proven its use in a real life exoskeleton.. Some earlier exoskeletons used pre-recorded natural gaits [3, 4, 6] but since these cannot be altered this was not an option for this study. To choose a gait generation algorithm the number of variables needed to define the gait was considered. This was important since on-line adaption of the gait was the main goal of this study. Every parameter that can be altered or is needed to define the gait would need a separate channel on the human machine interface. The algorithm that was used by IHMC and that is described in the paper was chosen because it only uses three parameters of which two are important for stability, step length and swing time. An added benefit is that these parameters are easy to explain and understand to and by subjects.

In order to find a set of default step parameters that result in a stable walking cycle for the exoskeleton a typical step length and step height for big steps in an exoskeleton were assumed (step length = 45 cm and step height = 20 cm). With these dimensions the swing time was tuned until a stable walking cycle was achieved.

The algorithm was adapted in such a way that it can handle re-initialization. This re-initialization was based on changes in step length and swing time. The changes in these variables were caused by a single variable called the combined input factor. This was done for two reasons, first of all this single factor can be changed independently of the hardware setup that is used. This allows for multiple possible hardware systems without making changes to the controller part of the software. If a setup is connected an inputHandler object is constructed (see section 2.4.10), this input handler will handle the interaction between the device and the combined input factor. This insures that no device can directly mess with the step parameters and ensures that the effect of all different hardware systems on the controller is the same.

A input factor was chosen over a direct increase of step length or swing time so that the default values of the gait can be altered without changing the relative magnitude of the human inputs. This ensured that the gait could be tuned to a stable gait without effecting the relative magnitude of the control possible by the human.

2.3. Three dimensional gaits

Since human beings when walking naturally make alterations to step width and step length in order to cope with disturbances [5] the exoskeleton controller was designed to be able to generate three dimensional gaits, including hip ab/adduction. this greatly increased complexity since the gait generation uses inverse kinematics to determine the joint angles at the way points. In order to maximize controller speed and decrease the complexity of the controller the hip ab/adducction was linearized. This drastically reduced the complexity of the inverse kinematics since this could be done numerically now in stead of with incremental methods.

When the simulation was finally connected to the hardware setup it became clear quickly that the proposed control strategy was so new that the domain of the research had to be narrowed down in order to keep it feasible. At this point the decision was made to investigate the human influence in a two dimensional gait properly instead of jumping to the more complex three dimensional gait. This is the reason that three dimensional gaits were supported by the controller but not used in the research.

2.4. Class overview

A dependancy UML of all classes in the controller project is shown in figure 2.1. All dependencies on classes from third party packages are not shown in this figure. In this section all classes will be briefly discussed to give an overview of the structure of the controller.

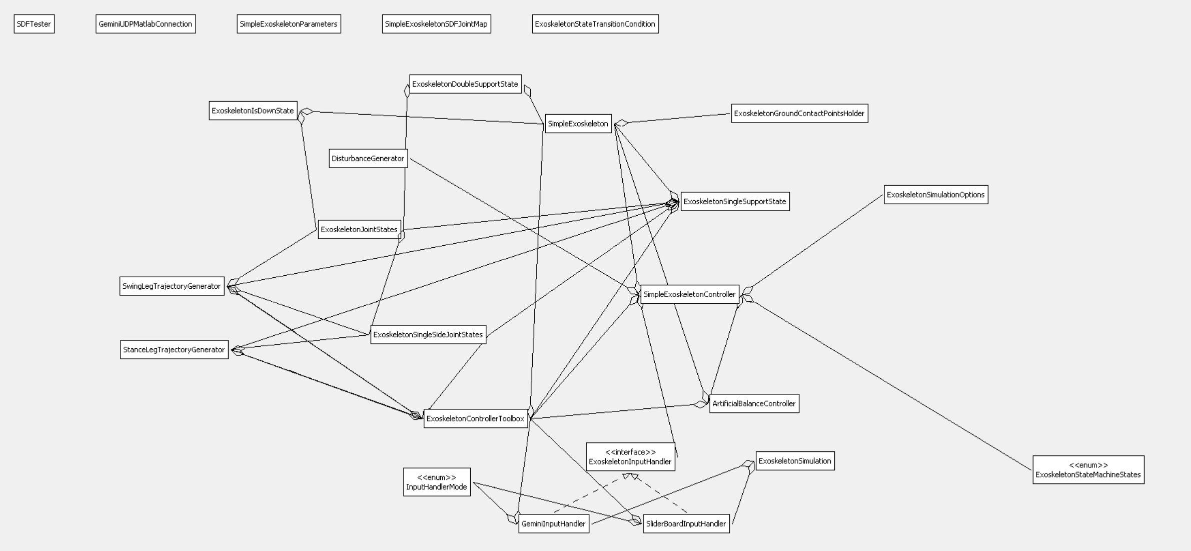
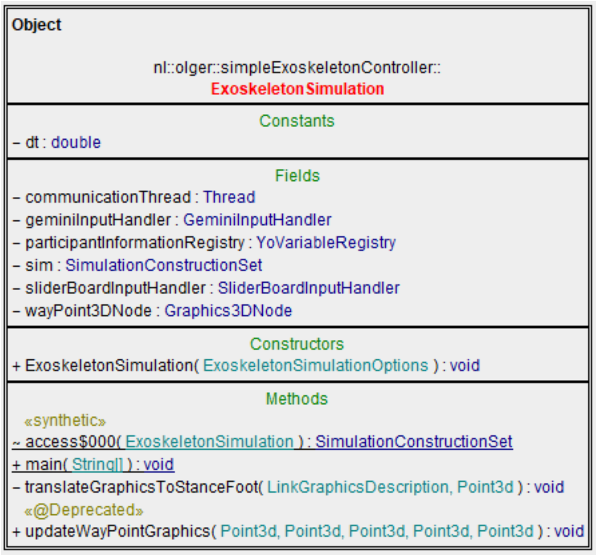

SDFTester
GeminiUDPMatlabConnection
SimpleExoskeletonParameters
SimpleExoskeletonSDFJointMap
ExoskeletonStateTransitionCondition
ExoskeletonDoubleSupportState
ExoskeletonIsDownState
SimpleExoskeleton
ExoskeletonGroundContactPointsHolder
DisturbanceGenerator
ExoskeletonSingleSupportState
ExoskeletonSimulationOptions
ExoskeletonJointStates
SwingLegTrajectoryGenerator
SimpleExoskeletonController
ExoskeletonSingleSideJointStates
StanceLegTrajectoryGenerator
ArtificialBalanceController
ExoskeletonControllerToolbox
<<interface>>
ExoskeletonInputHandler
ExoskeletonSimulation
<<enum>>
InputHandlerMode
<<enum>>
ExoskeletonStateMachineStates
GeminiInputHandler
SliderBoardInputHandler

Figure 2.1: A UML diagram of the dependencies of all classes in the `SimpleExoskeletonController` project. All dependencies on classes from other packages are left out. Including all classes from IHMC's Simulation Construction Set.

2.4.1. ExoskeletonSimulation

The main class in the project is the `ExoskeletonSimulation`, when this object is constructed a new simulation environment is created. This is a `SimulationConstructionSet` object but since that class belongs to an external package it is not visualized in figure 2.1. A UML diagram of `ExoskeletonSimulation` can be found in figure 2.2. In `ExoskeletonSimulation` instances of `SimpleExoskeleton` and `SimpleExoskeletonController` are created and attached to the simulation. An input handler is attached directly to the `ExoskeletonSimulation`. This can be either a `GeminiInputHandler` or a `SliderBoardInputHandler` which will be explained in the next subsections.

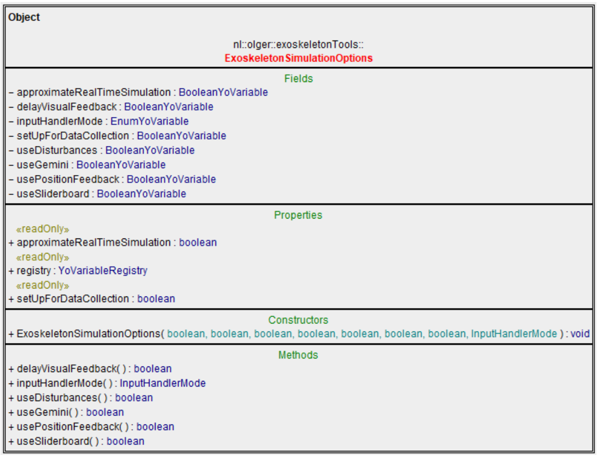
The deprecated method `updateWayPointGraphics` was used in an early phase of the project when the exoskeleton was suspended from the hips. This method would visualize the swing foot way points in the world frame. This method assumed that the hip position was equal to the world reference frame and thus stopped working properly when the exoskeleton started walking on the ground. At this point the method was deprecated and placed on a todo list. But since it was not essential it was never re-implemented.



```
Object

nl::olger::simpleExoskeletonController::
ExoskeletonSimulation

Constants
– dt : double

Fields
– communicationThread : Thread
– geminiInputHandler : GeminiInputHandler
– participantInformationRegistry : YoVariableRegistry
– sim : SimulationConstructionSet
– sliderBoardInputHandler : SliderBoardInputHandler
– wayPoint3DNode : Graphics3DNode

Constructors
+ ExoskeletonSimulation( ExoskeletonSimulationOptions ) : void

Methods
  «synthetic»
~ access$000( ExoskeletonSimulation ) : SimulationConstructionSet
+ main( String[] ) : void
– translateGraphicsToStanceFoot( LinkGraphicsDescription, Point3d ) : void
  «@Deprecated»
+ updateWayPointGraphics( Point3d, Point3d, Point3d, Point3d, Point3d ) : void
```

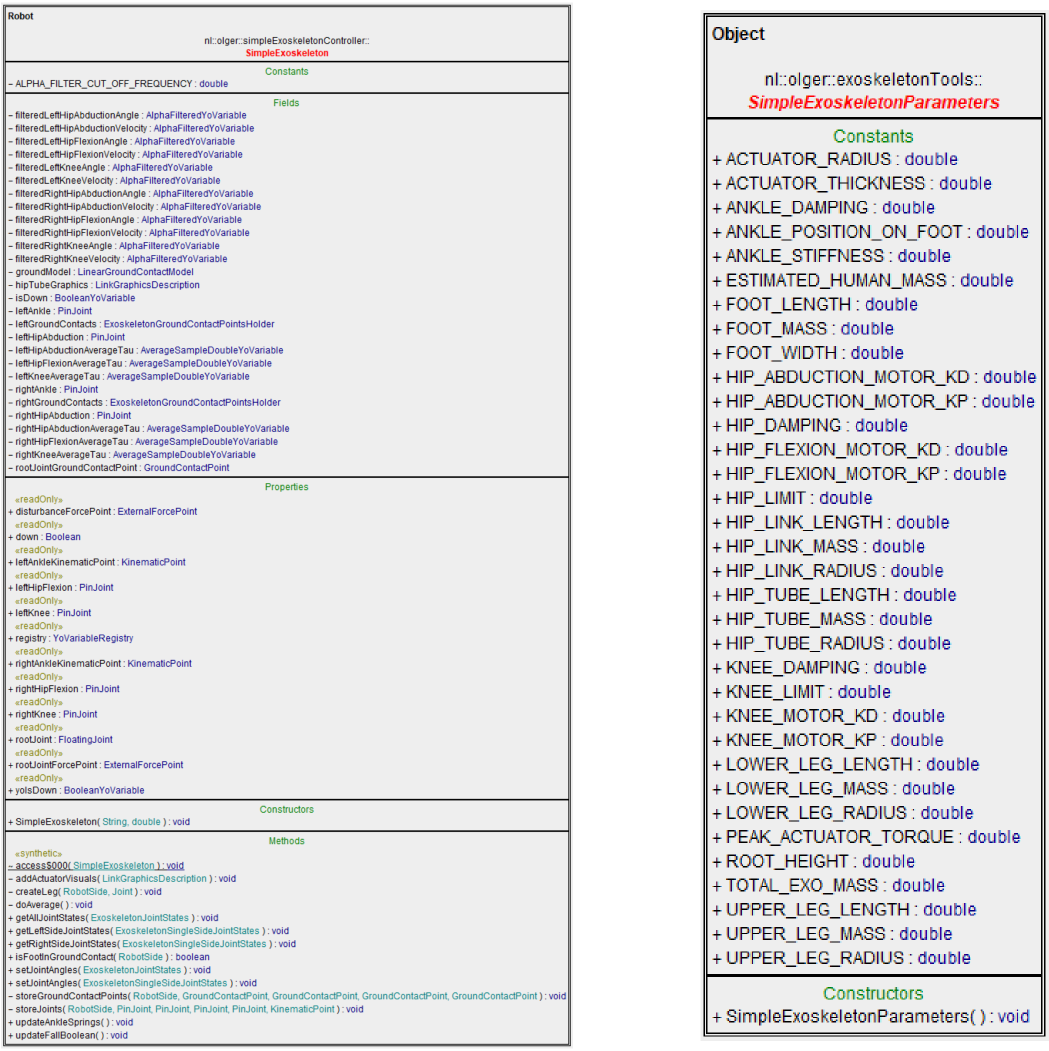
Figure 2.2: The UML diagram of `ExoskeletonSimulation`

When the `SimpleExoskeletonSimulation` is constructed an `ExoskeletonSimulationOptions` object is passed to the constructor. This object contains all relevant options for the different conditions of the experiment. This way it is easy to change these conditions and have an overview of the current settings. The UML diagram of the `ExoskeletonSimulationOptions` class can be found in figure 2.3



```
Object

nl::olger::exoskeletonTools::
ExoskeletonSimulationOptions

Fields
– approximateRealTimeSimulation : BooleanYoVariable
– delayVisualFeedback : BooleanYoVariable
– inputHandlerMode : EnumYoVariable
– setUpForDataCollection : BooleanYoVariable
– useDisturbances : BooleanYoVariable
– useGemini : BooleanYoVariable
– usePositionFeedback : BooleanYoVariable
– useSliderboard : BooleanYoVariable

Properties
  «readOnly»
+ approximateRealTimeSimulation : boolean
  «readOnly»
+ registry : YoVariableRegistry
  «readOnly»
+ setUpForDataCollection : boolean

Constructors
+ ExoskeletonSimulationOptions( boolean, boolean, boolean, boolean, boolean, boolean, boolean, InputHandlerMode ) : void

Methods
+ delayVisualFeedback( ) : boolean
+ inputHandlerMode( ) : InputHandlerMode
+ useDisturbances( ) : boolean
+ useGemini( ) : boolean
+ usePositionFeedback( ) : boolean
+ useSliderboard( ) : boolean
```

Figure 2.3: The UML diagram of the `ExoskeletonSimulationOptions` class



```
Robot

nl::olger::simpleExoskeletonController::
SimpleExoskeleton

Constants
– ALPHA_FILTER_CUT_OFF_FREQUENCY : double

Fields
– filteredLeftHipAbductionAngle : AlphaFilteredYoVariable
– filteredLeftHipAbductionVelocity : AlphaFilteredYoVariable
– filteredLeftHipFlexionAngle : AlphaFilteredYoVariable
– filteredLeftHipFlexionVelocity : AlphaFilteredYoVariable
– filteredLeftKneeAngle : AlphaFilteredYoVariable
– filteredLeftKneeVelocity : AlphaFilteredYoVariable
– filteredRightHipAbductionAngle : AlphaFilteredYoVariable
– filteredRightHipAbductionVelocity : AlphaFilteredYoVariable
– filteredRightHipFlexionAngle : AlphaFilteredYoVariable
– filteredRightHipFlexionVelocity : AlphaFilteredYoVariable
– filteredRightKneeAngle : AlphaFilteredYoVariable
– filteredRightKneeVelocity : AlphaFilteredYoVariable
– groundModel : LinearGroundContactModel
– hipTubeGraphics : LinkGraphicsDescription
– isDown : BooleanYoVariable
– leftAnkle : PinJoint
– leftGroundContacts : ExoskeletonGroundContactPointsHolder
– leftHipAbduction : PinJoint
– leftHipAbductionAverageTau : AverageSampleDoubleYoVariable
– leftHipFlexionAverageTau : AverageSampleDoubleYoVariable
– leftKneeAverageTau : AverageSampleDoubleYoVariable
– rightAnkle : PinJoint
– rightGroundContacts : ExoskeletonGroundContactPointsHolder
– rightHipAbduction : PinJoint
– rightHipAbductionAverageTau : AverageSampleDoubleYoVariable
– rightHipFlexionAverageTau : AverageSampleDoubleYoVariable
– rightKneeAverageTau : AverageSampleDoubleYoVariable
– rootJointGroundContactPoint : GroundContactPoint

Properties
  «readOnly»
+ disturbanceForcePoint : ExternalForcePoint
  «readOnly»
+ down : Boolean
  «readOnly»
+ leftAnkleKinematicPoint : KinematicPoint
  «readOnly»
+ leftHipFlexion : PinJoint
  «readOnly»
+ leftKnee : PinJoint
  «readOnly»
+ registry : YoVariableRegistry
  «readOnly»
+ rightAnkleKinematicPoint : KinematicPoint
  «readOnly»
+ rightHipFlexion : PinJoint
  «readOnly»
+ rightKnee : PinJoint
  «readOnly»
+ rootJoint : FloatingJoint
  «readOnly»
+ rootJointForcePoint : ExternalForcePoint
  «readOnly»
+ yoIsDown : BooleanYoVariable

Constructors
+ SimpleExoskeleton( String, double ) : void

Methods
  «synthetic»
~ access$000( SimpleExoskeleton ) : void
– addActuatorVisuals( LinkGraphicsDescription ) : void
– createLeg( RobotSide, Joint ) : void
– doAverage( ) : void
+ getAllJointStates( ExoskeletonJointStates ) : void
+ getLeftSideJointStates( ExoskeletonSingleSideJointStates ) : void
+ getRightSideJointStates( ExoskeletonSingleSideJointStates ) : void
+ isFootInGroundContact( RobotSide ) : boolean
+ setJointAngles( ExoskeletonJointStates ) : void
+ setJointAngles( ExoskeletonSingleSideJointStates ) : void
– storeGroundContactPoints( RobotSide, GroundContactPoint, GroundContactPoint, GroundContactPoint, GroundContactPoint ) : void
– storeJoints( RobotSide, PinJoint, PinJoint, PinJoint, PinJoint, KinematicPoint ) : void
+ updateAnkleSprings( ) : void
+ updateFallBoolean( ) : void
```

(a) The UML diagram of `SimpleExoskeleton`

```
Object

nl::olger::exoskeletonTools::
SimpleExoskeletonParameters

Constants
+ ACTUATOR_RADIUS : double
+ ACTUATOR_THICKNESS : double
+ ANKLE_DAMPING : double
+ ANKLE_POSITION_ON_FOOT : double
+ ANKLE_STIFFNESS : double
+ ESTIMATED_HUMAN_MASS : double
+ FOOT_LENGTH : double
+ FOOT_MASS : double
+ FOOT_WIDTH : double
+ HIP_ABDUCTION_MOTOR_KD : double
+ HIP_ABDUCTION_MOTOR_KP : double
+ HIP_DAMPING : double
+ HIP_FLEXION_MOTOR_KD : double
+ HIP_FLEXION_MOTOR_KP : double
+ HIP_LIMIT : double
+ HIP_LINK_LENGTH : double
+ HIP_LINK_MASS : double
+ HIP_LINK_RADIUS : double
+ HIP_TUBE_LENGTH : double
+ HIP_TUBE_MASS : double
+ HIP_TUBE_RADIUS : double
+ KNEE_DAMPING : double
+ KNEE_LIMIT : double
+ KNEE_MOTOR_KD : double
+ KNEE_MOTOR_KP : double
+ LOWER_LEG_LENGTH : double
+ LOWER_LEG_MASS : double
+ LOWER_LEG_RADIUS : double
+ PEAK_ACTUATOR_TORQUE : double
+ ROOT_HEIGHT : double
+ TOTAL_EXO_MASS : double
+ UPPER_LEG_LENGTH : double
+ UPPER_LEG_MASS : double
+ UPPER_LEG_RADIUS : double

Constructors
+ SimpleExoskeletonParameters( ) : void
```

(b) The UML Diagram of `SimpleExoskeletonParameters`

Figure 2.4: UML Diagrams of the robot

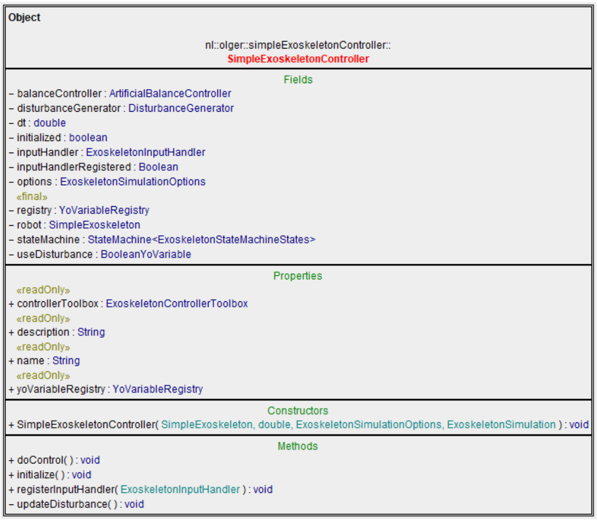
2.4.2. Robot and Controller

The robot used in this project is defined in the `SimpleExoskeleton` class, a UML diagram of this class can be found in figure 2.4a. The robot object holds all physical and visual information for the simulation. The parameters of the robot are stored in a separate static class called `SimpleExoskeletonParameters` (UML in figure 2.4b). Next to holding this information the robot class acts as a sensor reader and output writer. It passes information on the current state of the robot to the controller and writes position and velocity references to the robot.

The current state exists of all joint angles and velocities and the four ground contact sensors per foot. These ground contact sensors are only used to detect successful steps and are not used in the state machine. The references are converted to forces using a PD-controller which mimics a lower level motor amplifier control loop. A low-pass filter is added to all references as a safety feature to prevent peaks in the reference values. All gains and the filter frequency are given in the paper.

The `SimpleExoskeleton` will detect its own falls (when the hips hit the ground) and will update a PD controller on the ankle joints every controller tick which mimics a passive spring damper combination (parameters are given in the paper).

The `SimpleExoskeletonController` class is the main object of the actual controller, it's UML diagram can be found in figure 2.5. This controller calls all relevant other objects during every control tick. It first updates some variables in the `ControllerToolbox`, it then updates the `InputHandler`. After this the action of the current state machine state is called where the calculations of the reference positions and velocities



```
Object

nl::olger::simpleExoskeletonController::
SimpleExoskeletonController

Fields
– balanceController : ArtificialBalanceController
– disturbanceGenerator : DisturbanceGenerator
– dt : double
– initialized : boolean
– inputHandler : ExoskeletonInputHandler
– inputHandlerRegistered : Boolean
– options : ExoskeletonSimulationOptions
  «final»
– registry : YoVariableRegistry
– robot : SimpleExoskeleton
– stateMachine : StateMachine<ExoskeletonStateMachineStates>
– useDisturbance : BooleanYoVariable

Properties
  «readOnly»
+ controllerToolbox : ExoskeletonControllerToolbox
  «readOnly»
+ description : String
  «readOnly»
+ name : String
  «readOnly»
+ yoVariableRegistry : YoVariableRegistry

Constructors
+ SimpleExoskeletonController( SimpleExoskeleton, double, ExoskeletonSimulationOptions, ExoskeletonSimulation ) : void

Methods
+ doControl( ) : void
+ initialize( ) : void
+ registerInputHandler( ExoskeletonInputHandler ) : void
– updateDisturbance( ) : void
```

Figure 2.5: The UML diagram of the `SimpleExoskeletonController`

are made. Then it updates the ankle springs and the fall detection in the `SimpleExoskeleton`. And finally it updates the disturbances and the artificial balance controller. When all of these actions are finished it will check the state machines transition conditions and then make the thread sleep for the remaining time of the controller tick. This is done to approximate a real time simulation.

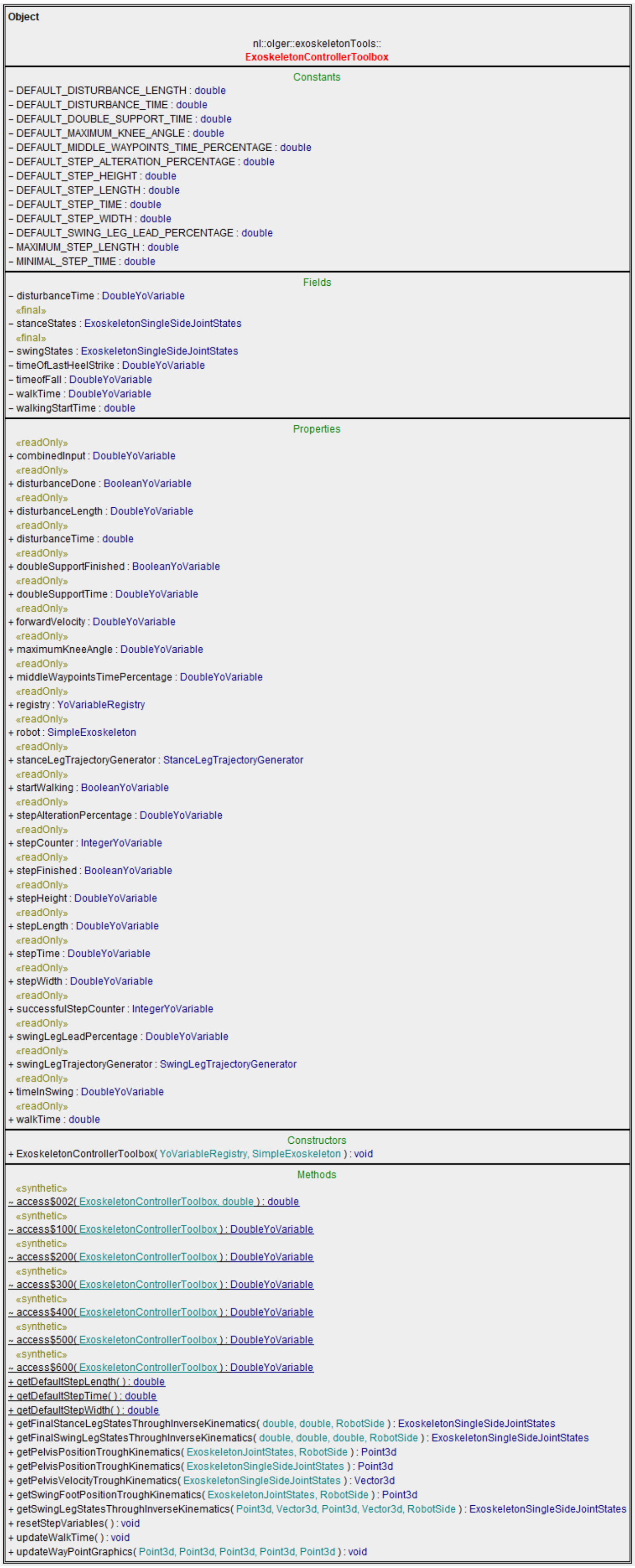
All of these different components will be discussed in the next subsections.

2.4.3. Controller Toolbox

The controller toolbox contains a lot of methods and variables that are used by the controller, state machine and trajectory generators. Its UML representation can be found in figure 2.6. It is constructed in the constructor of the controller and then passed to all objects that need access to these methods and variables. The `ExoskeletonControllerToolbox` holds all the methods to do (inverse-)kinematics needed in the trajectory generators. It also holds all step parameters and their default values. It holds the combinedInputFactor used by the input devices and it holds all Booleans that are used to determine if a state change should occur.

By storing all these shared variables and methods in one toolbox, only the toolbox has to be passed around and discussions about were a variable should live are settled. The added benefit is that this toolbox gives all the objects it is passed to access to the robot. So the robot object doesn't have to be passed around.

The step parameters that live in this class have variable changed listeners attached to them. These listeners call the re-initialization of the trajectory generators through the `ExoskeletonSingleSupportState` when the parameter is changed.



```
Object

nl::olger::exoskeletonTools::
ExoskeletonControllerToolbox

Constants
– DEFAULT_DISTURBANCE_LENGTH : double
– DEFAULT_DISTURBANCE_TIME : double
– DEFAULT_DOUBLE_SUPPORT_TIME : double
– DEFAULT_MAXIMUM_KNEE_ANGLE : double
– DEFAULT_MIDDLE_WAYPOINTS_TIME_PERCENTAGE : double
– DEFAULT_STEP_ALTERATION_PERCENTAGE : double
– DEFAULT_STEP_HEIGHT : double
– DEFAULT_STEP_LENGTH : double
– DEFAULT_STEP_TIME : double
– DEFAULT_STEP_WIDTH : double
– DEFAULT_SWING_LEG_LEAD_PERCENTAGE : double
– MAXIMUM_STEP_LENGTH : double
– MINIMAL_STEP_TIME : double

Fields
– disturbanceTime : DoubleYoVariable
  «final»
– stanceStates : ExoskeletonSingleSideJointStates
  «final»
– swingStates : ExoskeletonSingleSideJointStates
– timeOfLastHeelStrike : DoubleYoVariable
– timeofFall : DoubleYoVariable
– walkTime : DoubleYoVariable
– walkingStartTime : double

Properties
  «readOnly»
+ combinedInput : DoubleYoVariable
  «readOnly»
+ disturbanceDone : BooleanYoVariable
  «readOnly»
+ disturbanceLength : DoubleYoVariable
  «readOnly»
+ disturbanceTime : double
  «readOnly»
+ doubleSupportFinished : BooleanYoVariable
  «readOnly»
+ doubleSupportTime : DoubleYoVariable
  «readOnly»
+ forwardVelocity : DoubleYoVariable
  «readOnly»
+ maximumKneeAngle : DoubleYoVariable
  «readOnly»
+ middleWaypointsTimePercentage : DoubleYoVariable
  «readOnly»
+ registry : YoVariableRegistry
  «readOnly»
+ robot : SimpleExoskeleton
  «readOnly»
+ stanceLegTrajectoryGenerator : StanceLegTrajectoryGenerator
  «readOnly»
+ startWalking : BooleanYoVariable
  «readOnly»
+ stepAlterationPercentage : DoubleYoVariable
  «readOnly»
+ stepCounter : IntegerYoVariable
  «readOnly»
+ stepFinished : BooleanYoVariable
  «readOnly»
+ stepHeight : DoubleYoVariable
  «readOnly»
+ stepLength : DoubleYoVariable
  «readOnly»
+ stepTime : DoubleYoVariable
  «readOnly»
+ stepWidth : DoubleYoVariable
  «readOnly»
+ successfulStepCounter : IntegerYoVariable
  «readOnly»
+ swingLegLeadPercentage : DoubleYoVariable
  «readOnly»
+ swingLegTrajectoryGenerator : SwingLegTrajectoryGenerator
  «readOnly»
+ timeInSwing : DoubleYoVariable
  «readOnly»
+ walkTime : double

Constructors
+ ExoskeletonControllerToolbox( YoVariableRegistry, SimpleExoskeleton ) : void

Methods
  «synthetic»
~ access$002( ExoskeletonControllerToolbox, double ) : double
  «synthetic»
~ access$100( ExoskeletonControllerToolbox ) : DoubleYoVariable
  «synthetic»
~ access$200( ExoskeletonControllerToolbox ) : DoubleYoVariable
  «synthetic»
~ access$300( ExoskeletonControllerToolbox ) : DoubleYoVariable
  «synthetic»
~ access$400( ExoskeletonControllerToolbox ) : DoubleYoVariable
  «synthetic»
~ access$500( ExoskeletonControllerToolbox ) : DoubleYoVariable
  «synthetic»
~ access$600( ExoskeletonControllerToolbox ) : DoubleYoVariable
+ getDefaultStepLength( ) : double
+ getDefaultStepTime( ) : double
+ getDefaultStepWidth( ) : double
+ getFinalStanceLegStatesThroughInverseKinematics( double, double, RobotSide ) : ExoskeletonSingleSideJointStates
+ getFinalSwingLegStatesThroughInverseKinematics( double, double, double, RobotSide ) : ExoskeletonSingleSideJointStates
+ getPelvisPositionTroughKinematics( ExoskeletonJointStates, RobotSide ) : Point3d
+ getPelvisPositionTroughKinematics( ExoskeletonSingleSideJointStates ) : Point3d
+ getPelvisVelocityTroughKinematics( ExoskeletonSingleSideJointStates ) : Vector3d
+ getSwingFootPositionTroughKinematics( ExoskeletonJointStates, RobotSide ) : Point3d
+ getSwingLegStatesThroughInverseKinematics( Point3d, Vector3d, Point3d, Vector3d, RobotSide ) : ExoskeletonSingleSideJointStates
+ resetStepVariables( ) : void
+ updateWalkTime( ) : void
+ updateWayPointGraphics( Point3d, Point3d, Point3d, Point3d, Point3d ) : void
```

Figure 2.6: The UML diagram of the `ExoskeletonControllerToolbox`

2.4.4. Input Handlers

The controller is setup in such a way that multiple input devices can be used with the same controller, this is done with the `InputHandler` interface. The UML diagram of the `InputHandler` can be found in figure 2.7. Two hardware specific input handlers implement this interface, the `GeminiInputHandler` (figure 2.8a) and the `SliderBoardInputHandler` (figure 2.8b). An Enum was created to switch between swing time and step length control, the `InputHandlerMode` (figure 2.9)

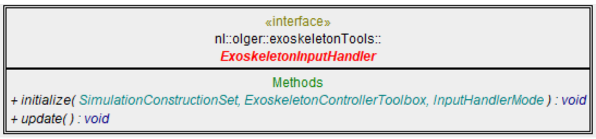
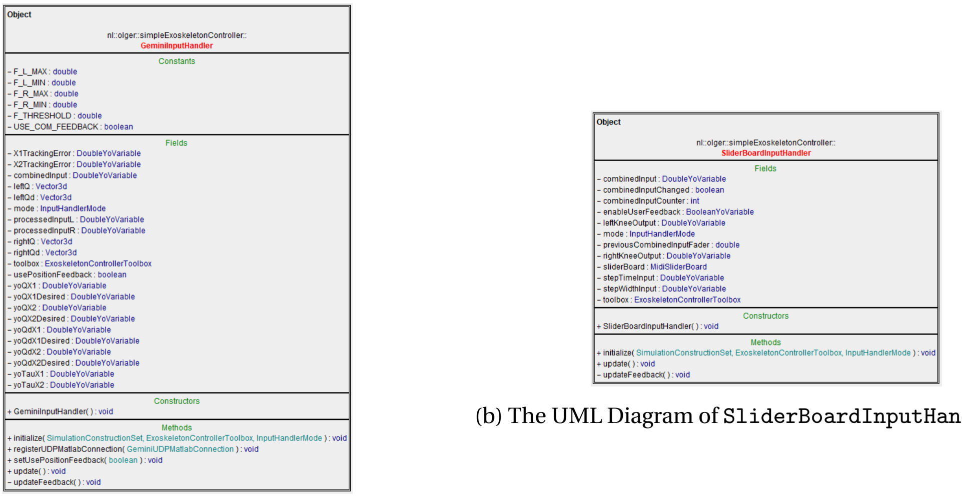

«interface»
nl::olger::exoskeletonTools::
ExoskeletonInputHandler
Methods
+ initialize(SimulationConstructionSet, ExoskeletonControllerToolbox, InputHandlerMode) : void
+ update( ) : void

Figure 2.7: The UML diagram of the `ExoskeletonInputHandler` interface


Object
nl::olger::simpleExoskeletonController::
GeminiInputHandler
Constants
– F_L_MAX : double
– F_L_MIN : double
– F_R_MAX : double
– F_R_MIN : double
– F_THRESHOLD : double
– USE_COM_FEEDBACK : boolean
Fields
– X1TrackingError : DoubleYoVariable
– X2TrackingError : DoubleYoVariable
– combinedInput : DoubleYoVariable
– leftQ : Vector3d
– leftQd : Vector3d
– mode : InputHandlerMode
– processedInputL : DoubleYoVariable
– processedInputR : DoubleYoVariable
– rightQ : Vector3d
– rightQd : Vector3d
– toolbox : ExoskeletonControllerToolbox
– usePositionFeedback : boolean
– yoQX1 : DoubleYoVariable
– yoQX1Desired : DoubleYoVariable
– yoQX2 : DoubleYoVariable
– yoQX2Desired : DoubleYoVariable
– yoQdX1 : DoubleYoVariable
– yoQdX1Desired : DoubleYoVariable
– yoQdX2 : DoubleYoVariable
– yoQdX2Desired : DoubleYoVariable
– yoTauX1 : DoubleYoVariable
– yoTauX2 : DoubleYoVariable
Constructors
+ GeminiInputHandler( ) : void
Methods
+ initialize(SimulationConstructionSet, ExoskeletonControllerToolbox, InputHandlerMode) : void
+ registerUDPMatlabConnection(GeminiUDPMatlabConnection) : void
+ setUsePositionFeedback(boolean) : void
+ update( ) : void
– updateFeedback( ) : void
Object
nl::olger::simpleExoskeletonController::
SliderBoardInputHandler
Fields
– combinedInput : DoubleYoVariable
– combinedInputChanged : boolean
– combinedInputCounter : int
– enableUserFeedback : BooleanYoVariable
– leftKneeOutput : DoubleYoVariable
– mode : InputHandlerMode
– previousCombinedInputFader : double
– rightKneeOutput : DoubleYoVariable
– sliderBoard : MidiSliderBoard
– stepTimeInput : DoubleYoVariable
– stepWidthInput : DoubleYoVariable
– toolbox : ExoskeletonControllerToolbox
Constructors
+ SliderBoardInputHandler( ) : void
Methods
+ initialize(SimulationConstructionSet, ExoskeletonControllerToolbox, InputHandlerMode) : void
+ update( ) : void
– updateFeedback( ) : void

(b) The UML Diagram of `SliderBoardInputHandler`

(a) The UML diagram of `GeminiInputhandler`

Figure 2.8: UML Diagrams of the Input Handlers

When comparing the two device specific input handlers it immidiatly stands out that the Gemini input handler is somewhat more complex then the slider board version. This can be explained by the fact that the slider board is a standard functionality in SCS which uses a midi controller with moterized faders. The options of this slider board are rather limited but it is easy to read and write the positions of faders.

The Gemini joystick setup is the setup that was eventually used in the experiments. Since this was a device build by TU Delft there was no standard build in functionality. So all the filtering and scaling of the raw data had to be done in the input handler. Another difference is the fact that the `sliderBoard` object that handles the connection is a part of SCS. The connection with the gemini device ran through a UDP router in Matlab. This required the `GeminiUDPMatlabConnection` object (figure 2.10).

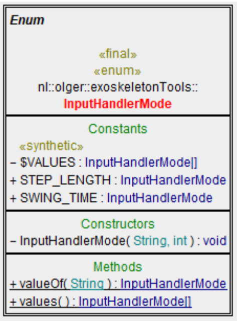
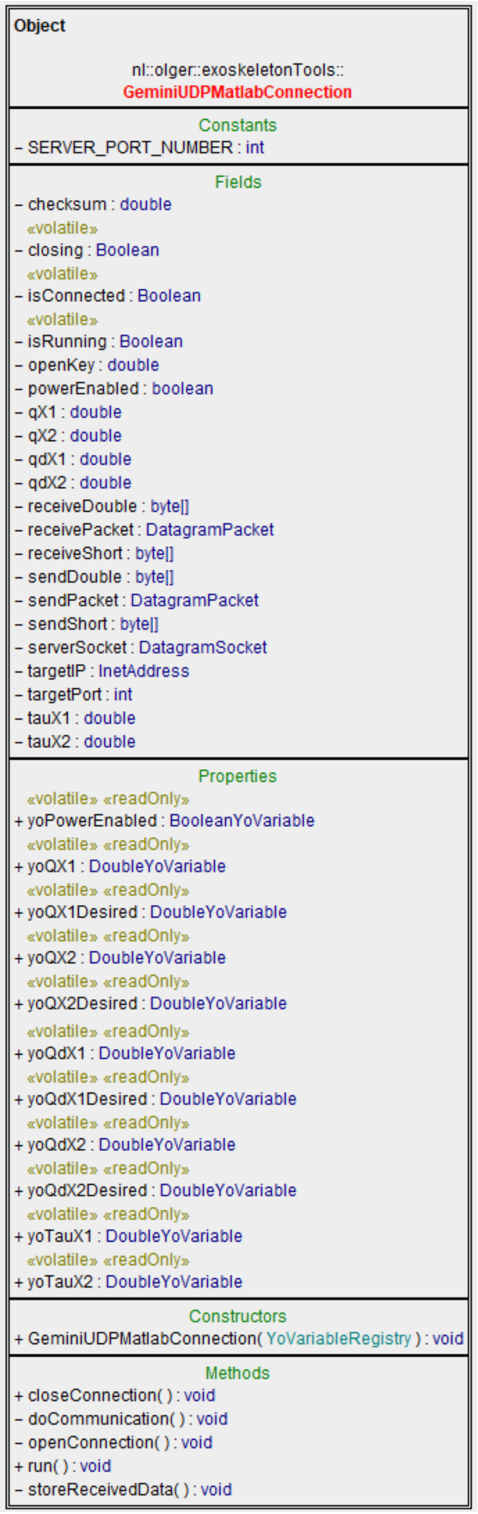

Enum
«final»
«enum»
nl::olger::exoskeletonTools::
InputHandlerMode
Constants
«synthetic»
– $VALUES : InputHandlerMode[]
+ STEP_LENGTH : InputHandlerMode
+ SWING_TIME : InputHandlerMode
Constructors
– InputHandlerMode(String, int) : void
Methods
+ valueOf(String) : InputHandlerMode
+ values( ) : InputHandlerMode[]

Figure 2.9: The UML diagram of the `InputHandlerMode` Enum

This object connects to Matlab over a UDP connection with the localhost. It uses a handshake as the first message and an opening key and checksum system with all other messages to prevent skew messages caused by lost packets. All properties of this class are volatile because the communication runs in a separate thread from the simulation. The reason why this UDP connection was chosen over a direct TCP connection with the Bachmann controller can be found in chapter 3.

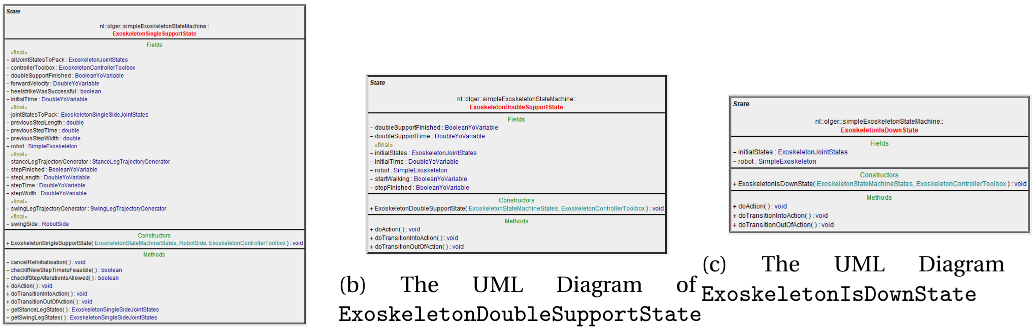


```
Object

nl::olger::exoskeletonTools::
GeminiUDPMatlabConnection

Constants
– SERVER_PORT_NUMBER : int

Fields
– checksum : double
  «volatile»
– closing : Boolean
  «volatile»
– isConnected : Boolean
  «volatile»
– isRunning : Boolean
– openKey : double
– powerEnabled : boolean
– qX1 : double
– qX2 : double
– qdX1 : double
– qdX2 : double
– receiveDouble : byte[]
– receivePacket : DatagramPacket
– receiveShort : byte[]
– sendDouble : byte[]
– sendPacket : DatagramPacket
– sendShort : byte[]
– serverSocket : DatagramSocket
– targetIP : InetAddress
– targetPort : int
– tauX1 : double
– tauX2 : double

Properties
  «volatile» «readOnly»
+ yoPowerEnabled : BooleanYoVariable
  «volatile» «readOnly»
+ yoQX1 : DoubleYoVariable
  «volatile» «readOnly»
+ yoQX1Desired : DoubleYoVariable
  «volatile» «readOnly»
+ yoQX2 : DoubleYoVariable
  «volatile» «readOnly»
+ yoQX2Desired : DoubleYoVariable
  «volatile» «readOnly»
+ yoQdX1 : DoubleYoVariable
  «volatile» «readOnly»
+ yoQdX1Desired : DoubleYoVariable
  «volatile» «readOnly»
+ yoQdX2 : DoubleYoVariable
  «volatile» «readOnly»
+ yoQdX2Desired : DoubleYoVariable
  «volatile» «readOnly»
+ yoTauX1 : DoubleYoVariable
  «volatile» «readOnly»
+ yoTauX2 : DoubleYoVariable

Constructors
+ GeminiUDPMatlabConnection( YoVariableRegistry ) : void

Methods
+ closeConnection( ) : void
– doCommunication( ) : void
– openConnection( ) : void
+ run( ) : void
– storeReceivedData( ) : void
```

Figure 2.10: The UML diagram of `GeminiUPDMatlabConnection`

2.4.5. State Machine

The state machine of the exoskeleton controller consistes of three state classes: `ExoskeletonSingleSupportState`, `ExoskeletonDoubleSupportState` and `ExoskeletonIsDownState`. Their UML diagrams can be found in figure 2.11



```
State

nl::olger::simpleExoskeletonStateMachine::
ExoskeletonSingleSupportState
Fields
  «final»
– allJointStatesToPack : ExoskeletonJointStates
– controllerToolbox : ExoskeletonControllerToolbox
– doubleSupportFinished : BooleanYoVariable
– forwardVelocity : DoubleYoVariable
– heelstrikeWasSuccessful : boolean
– initialTime : DoubleYoVariable
  «final»
– jointStatesToPack : ExoskeletonSingleSideJointStates
– previousStepLength : double
– previousStepTime : double
– previousStepWidth : double
– robot : SimpleExoskeleton
  «final»
– stanceLegTrajectoryGenerator : StanceLegTrajectoryGenerator
– stepFinished : BooleanYoVariable
– stepLength : DoubleYoVariable
– stepTime : DoubleYoVariable
– stepWidth : DoubleYoVariable
  «final»
– swingLegTrajectoryGenerator : SwingLegTrajectoryGenerator
  «final»
– swingSide : RobotSide
Constructors
+ ExoskeletonSingleSupportState( ExoskeletonStateMachineStates, RobotSide, ExoskeletonControllerToolbox ) : void
Methods
– cancelReInitialisation( ) : void
– checkIfNewStepTimeIsFeasible( ) : boolean
– checkIfStepAlterationIsAllowed( ) : boolean
+ doAction( ) : void
+ doTransitionIntoAction( ) : void
+ doTransitionOutOfAction( ) : void
– getStanceLegStates( ) : ExoskeletonSingleSideJointStates
– getSwingLegStates( ) : ExoskeletonSingleSideJointStates
```

(a) The UML diagram of `ExoskeletonSingleSupportState`

```
State

nl::olger::simpleExoskeletonStateMachine::
ExoskeletonDoubleSupportState
Fields
– doubleSupportFinished : BooleanYoVariable
– doubleSupportTime : DoubleYoVariable
  «final»
– initialStates : ExoskeletonJointStates
– initialTime : DoubleYoVariable
– robot : SimpleExoskeleton
– startWalking : BooleanYoVariable
– stepFinished : BooleanYoVariable
Constructors
+ ExoskeletonDoubleSupportState( ExoskeletonStateMachineStates, ExoskeletonControllerToolbox ) : void
Methods
+ doAction( ) : void
+ doTransitionIntoAction( ) : void
+ doTransitionOutOfAction( ) : void
```

(b) The UML Diagram of `ExoskeletonDoubleSupportState`

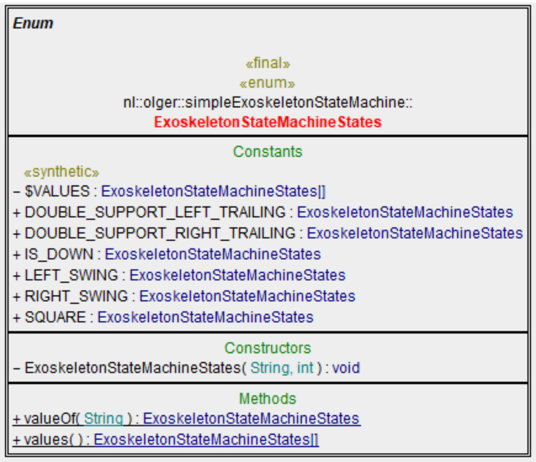
```
State

nl::olger::simpleExoskeletonStateMachine::
ExoskeletonIsDownState
Fields
– initialStates : ExoskeletonJointStates
– robot : SimpleExoskeleton
Constructors
+ ExoskeletonIsDownState( ExoskeletonStateMachineStates, ExoskeletonControllerToolbox ) : void
Methods
+ doAction( ) : void
+ doTransitionIntoAction( ) : void
+ doTransitionOutOfAction( ) : void
```

(c) The UML Diagram of `ExoskeletonIsDownState`

Figure 2.11: UML Diagrams of the States

At the initialization of the state machine two `ExoskeletonSingleSupportState` objects are made, one for the left and one for the right swing phase. Three versions of `ExoskeletonDoubleSupportState` are constructed, one for the right foot trailing, one for the left foot trailing and one for the squared position. And one object from the `ExoskeletonIsDownState`. All of these states have a `doAction` method, the controller will call the `doAction` method from the current state every controller tick. This method will then calculate the joint reference values and write them to the robot. When a new state is entered the state will call the appropriate trajectory generators to initialize.

The states hold an `ExoskeletonStates` Enum (figure 2.12) which diffrentiates between the right and left version of all states. This way the joint references are written to the correct leg. The `ExoskeletonStateTransitionConditi` object holds only one Boolean and triggers the state transition when this Boolean is true. Because it only holds one Boolean there is no UML representation presented here. It is a separate object so more complex state transition conditions could be used. The state machine object itself is part of the SCS package.

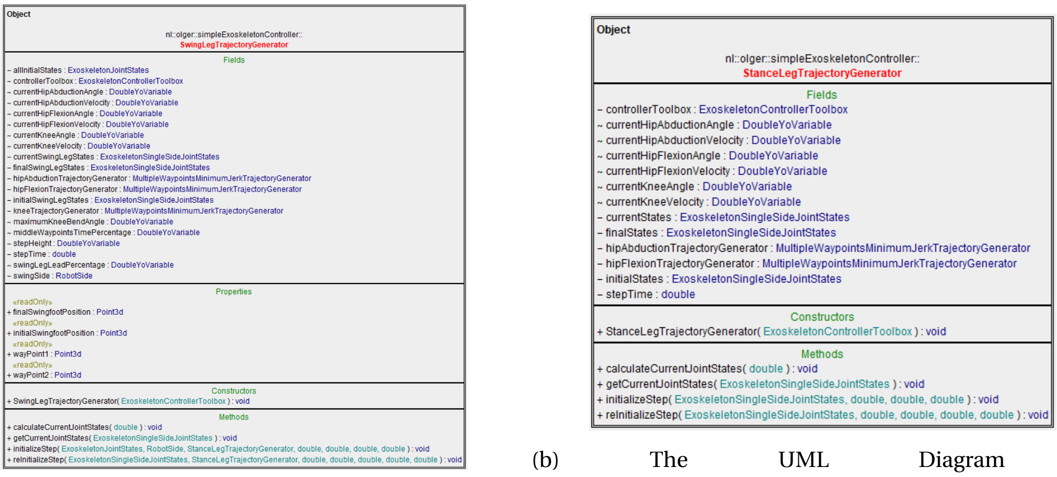


```
Enum

«final»
«enum»
nl::olger::simpleExoskeletonStateMachine::
ExoskeletonStateMachineStates
Constants
  «synthetic»
– $VALUES : ExoskeletonStateMachineStates[]
+ DOUBLE_SUPPORT_LEFT_TRAILING : ExoskeletonStateMachineStates
+ DOUBLE_SUPPORT_RIGHT_TRAILING : ExoskeletonStateMachineStates
+ IS_DOWN : ExoskeletonStateMachineStates
+ LEFT_SWING : ExoskeletonStateMachineStates
+ RIGHT_SWING : ExoskeletonStateMachineStates
+ SQUARE : ExoskeletonStateMachineStates
Constructors
– ExoskeletonStateMachineStates( String, int ) : void
Methods
+ valueOf( String ) : ExoskeletonStateMachineStates
+ values( ) : ExoskeletonStateMachineStates[]
```

Figure 2.12: The UML diagram for `ExoskeletonStateMachineStates`

2.4.6. Trajectory Generators

The `SwingLegTrajectoryGenerator` and `StanceLegTrajectoryGenerator` (UML diagrams in figure 2.13) generate the reference trajectories for all joints based on a way point algorithm. The working of this algorithm is explained in the scientific paper. The minimum jerk trajectory generators used by these trajectory generators are part of the open robotics software by IHMC.



Object

nl::olger::simpleExoskeletonController::
SwingLegTrajectoryGenerator

Fields
– allInitialStates : ExoskeletonJointStates
– controllerToolbox : ExoskeletonControllerToolbox
~ currentHipAbductionAngle : DoubleYoVariable
~ currentHipAbductionVelocity : DoubleYoVariable
~ currentHipFlexionAngle : DoubleYoVariable
~ currentHipFlexionVelocity : DoubleYoVariable
~ currentKneeAngle : DoubleYoVariable
~ currentKneeVelocity : DoubleYoVariable
– currentSwingLegStates : ExoskeletonSingleSideJointStates
– finalSwingLegStates : ExoskeletonSingleSideJointStates
– hipAbductionTrajectoryGenerator : MultipleWaypointsMinimumJerkTrajectoryGenerator
– hipFlexionTrajectoryGenerator : MultipleWaypointsMinimumJerkTrajectoryGenerator
– initialSwingLegStates : ExoskeletonSingleSideJointStates
– kneeTrajectoryGenerator : MultipleWaypointsMinimumJerkTrajectoryGenerator
~ maximumKneeBendAngle : DoubleYoVariable
~ middleWaypointsTimePercentage : DoubleYoVariable
~ stepHeight : DoubleYoVariable
– stepTime : double
~ swingLegLeadPercentage : DoubleYoVariable
– swingSide : RobotSide

Properties
«readOnly»
+ finalSwingfootPosition : Point3d
«readOnly»
+ initialSwingfootPosition : Point3d
«readOnly»
+ wayPoint1 : Point3d
«readOnly»
+ wayPoint2 : Point3d

Constructors
+ SwingLegTrajectoryGenerator(ExoskeletonControllerToolbox) : void

Methods
+ calculateCurrentJointStates(double) : void
+ getCurrentJointStates(ExoskeletonSingleSideJointStates) : void
+ initializeStep(ExoskeletonJointStates, RobotSide, StanceLegTrajectoryGenerator, double, double, double, double) : void
+ reInitializeStep(ExoskeletonSingleSideJointStates, StanceLegTrajectoryGenerator, double, double, double, double, double) : void

(a) The UML diagram of `SwingLegTrajectoryGenerator`

Object

nl::olger::simpleExoskeletonController::
StanceLegTrajectoryGenerator

Fields
– controllerToolbox : ExoskeletonControllerToolbox
~ currentHipAbductionAngle : DoubleYoVariable
~ currentHipAbductionVelocity : DoubleYoVariable
~ currentHipFlexionAngle : DoubleYoVariable
~ currentHipFlexionVelocity : DoubleYoVariable
~ currentKneeAngle : DoubleYoVariable
~ currentKneeVelocity : DoubleYoVariable
– currentStates : ExoskeletonSingleSideJointStates
– finalStates : ExoskeletonSingleSideJointStates
– hipAbductionTrajectoryGenerator : MultipleWaypointsMinimumJerkTrajectoryGenerator
– hipFlexionTrajectoryGenerator : MultipleWaypointsMinimumJerkTrajectoryGenerator
– initialStates : ExoskeletonSingleSideJointStates
– stepTime : double

Constructors
+ StanceLegTrajectoryGenerator(ExoskeletonControllerToolbox) : void

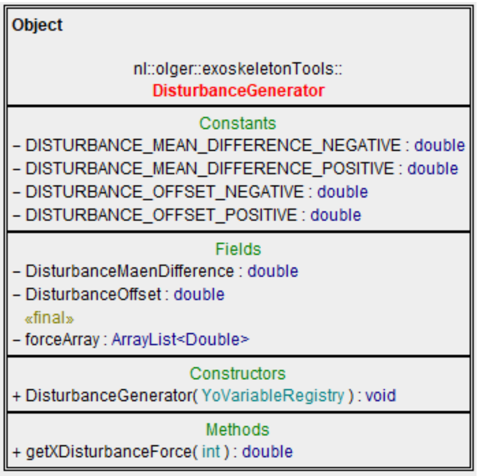
Methods
+ calculateCurrentJointStates(double) : void
+ getCurrentJointStates(ExoskeletonSingleSideJointStates) : void
+ initializeStep(ExoskeletonSingleSideJointStates, double, double, double) : void
+ reInitializeStep(ExoskeletonSingleSideJointStates, double, double, double, double) : void

(b) The UML Diagram of `StanceLegTrajectoryGenerator`

Figure 2.13: UML Diagrams of the Trjectory Generators

2.4.7. Disturbance and Balance

The working of the `DisturbanceGenerator` is explained in the scientific paper. A UML diagram is presented here in figure 2.14



Object

nl::olger::exoskeletonTools::
DisturbanceGenerator

Constants
– DISTURBANCE_MEAN_DIFFERENCE_NEGATIVE : double
– DISTURBANCE_MEAN_DIFFERENCE_POSITIVE : double
– DISTURBANCE_OFFSET_NEGATIVE : double
– DISTURBANCE_OFFSET_POSITIVE : double

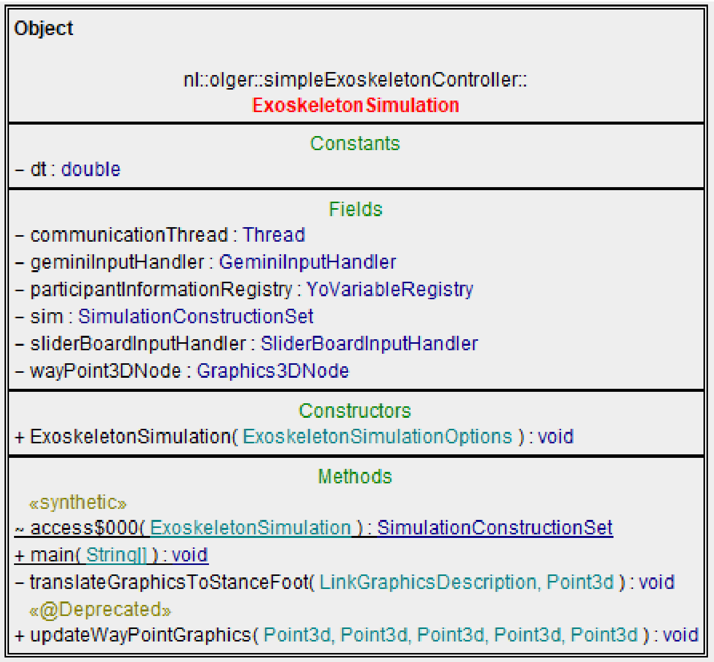
Fields
– DisturbanceMaenDifference : double
– DisturbanceOffset : double
«final»
– forceArray : ArrayList<Double>

Constructors
+ DisturbanceGenerator(YoVariableRegistry) : void

Methods
+ getXDisturbanceForce(int) : double

Figure 2.14: The UML diagram of the `DisturbanceGenerator`

The `ArtificialBalanceController` is also explained in the scientific paper although it in not explicitly mentioned. Its UML diagram can be found in figure 2.15. This class handles all additional balancing of the exoskeleton. So it keep it stable in the saggital plane, prevents the exoskeleton from turning and stabelizes the robot during the first two steps.



Object

nl::olger::simpleExoskeletonController::
ExoskeletonSimulation

Constants
- dt : double

Fields
- communicationThread : Thread
- geminiInputHandler : GeminiInputHandler
- participantInformationRegistry : YoVariableRegistry
- sim : SimulationConstructionSet
- sliderBoardInputHandler : SliderBoardInputHandler
- wayPoint3DNode : Graphics3DNode

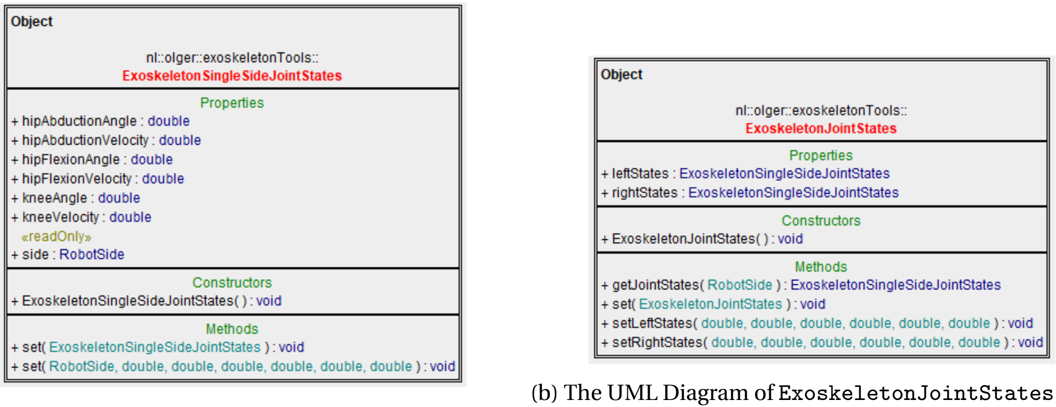
Constructors
+ ExoskeletonSimulation(ExoskeletonSimulationOptions) : void

Methods
«synthetic»
~ access$000(ExoskeletonSimulation) : SimulationConstructionSet
+ main(String[]) : void
- translateGraphicsToStanceFoot(LinkGraphicsDescription, Point3d) : void
«@Deprecated»
+ updateWayPointGraphics(Point3d, Point3d, Point3d, Point3d, Point3d) : void

Figure 2.15: The UML diagram of the `ArtificialBalanceController`

2.4.8. Joint State Containers

In order to easily pass the state of the complete exoskeleton or the state of one leg around, joint state containers were designed. Their UML diagrams can be found in figure 2.16



Object

nl::olger::exoskeletonTools::
ExoskeletonSingleSideJointStates

Properties
+ hipAbductionAngle : double
+ hipAbductionVelocity : double
+ hipFlexionAngle : double
+ hipFlexionVelocity : double
+ kneeAngle : double
+ kneeVelocity : double
«readOnly»
+ side : RobotSide

Constructors
+ ExoskeletonSingleSideJointStates( ) : void

Methods
+ set(ExoskeletonSingleSideJointStates) : void
+ set(RobotSide, double, double, double, double, double, double) : void

(a) The UML diagram of `ExoskeletonSingleSideJointStates`

Object

nl::olger::exoskeletonTools::
ExoskeletonJointStates

Properties
+ leftStates : ExoskeletonSingleSideJointStates
+ rightStates : ExoskeletonSingleSideJointStates

Constructors
+ ExoskeletonJointStates( ) : void

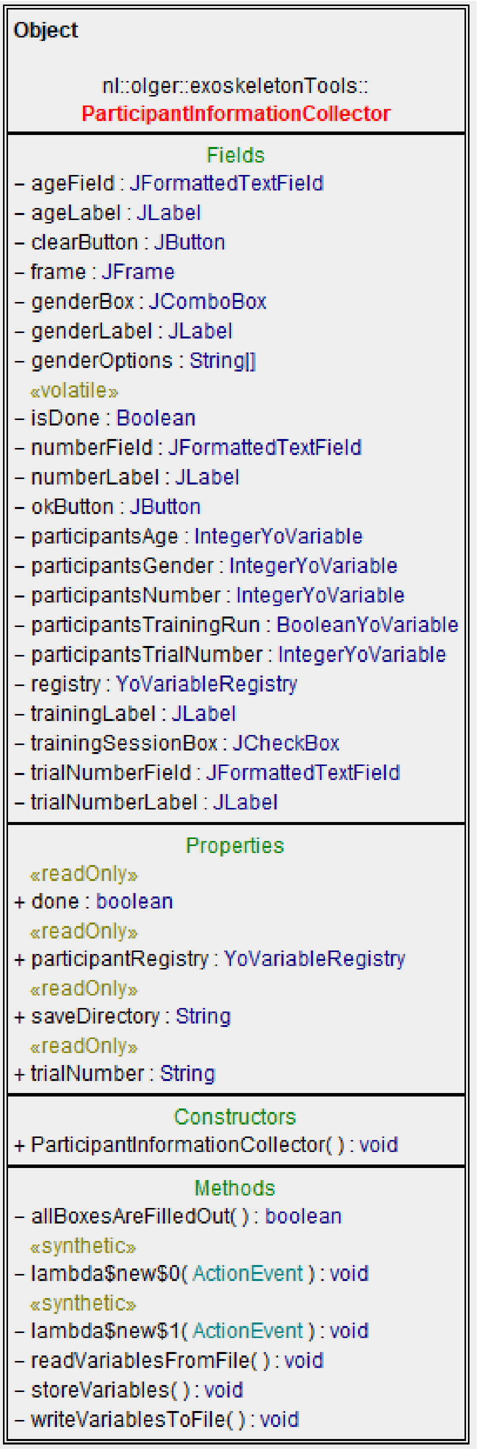
Methods
+ getJointStates(RobotSide) : ExoskeletonSingleSideJointStates
+ set(ExoskeletonJointStates) : void
+ setLeftStates(double, double, double, double, double, double) : void
+ setRightStates(double, double, double, double, double, double) : void

(b) The UML Diagram of `ExoskeletonJointStates`

Figure 2.16: UML Diagrams of the Joint State Containers

2.4.9. Participant Information Collector

The `ParticipantInformationCollector` is a GUI that stores a participant number, age, gender and a trial number in a `YoVaribleRegistry` this registry is than added to the simulation and thus saved with the data. It also overwrites a temporary text file containing the same information in order to automatically copy the data from the last run and increment the trialnumber. A UML diagram of this class can be found in figure 2.17.



Object

nl::olger::exoskeletonTools::
ParticipantInformationCollector

Fields
– ageField : JFormattedTextField
– ageLabel : JLabel
– clearButton : JButton
– frame : JFrame
– genderBox : JComboBox
– genderLabel : JLabel
– genderOptions : String[]
«volatile»
– isDone : Boolean
– numberField : JFormattedTextField
– numberLabel : JLabel
– okButton : JButton
– participantsAge : IntegerYoVariable
– participantsGender : IntegerYoVariable
– participantsNumber : IntegerYoVariable
– participantsTrainingRun : BooleanYoVariable
– participantsTrialNumber : IntegerYoVariable
– registry : YoVariableRegistry
– trainingLabel : JLabel
– trainingSessionBox : JCheckBox
– trialNumberField : JFormattedTextField
– trialNumberLabel : JLabel

Properties
«readOnly»
+ done : boolean
«readOnly»
+ participantRegistry : YoVariableRegistry
«readOnly»
+ saveDirectory : String
«readOnly»
+ trialNumber : String

Constructors
+ ParticipantInformationCollector( ) : void

Methods
– allBoxesAreFilledOut( ) : boolean
«synthetic»
– lambda$new$0(ActionEvent) : void
«synthetic»
– lambda$new$1(ActionEvent) : void
– readVariablesFromFile( ) : void
– storeVariables( ) : void
– writeVariablesToFile( ) : void

Figure 2.17: The UML diagram of the `ParticipantInformationCollector`

2.4.10. SDF

The classes `SDFTester` and `SimpleExoskeletonSDFJointMap` were used to make and test a SDF version of the robot model, which was first only defined by the class `SimpleExoskeleton`. This was done in order to enable the robot model to be send to a remote visualizer over a TCP-connection. But as said before, the remote visualizer functionality of SCS was not used in the final experiment so these classes are somewhat redundant.

3 Device

The hardware used in the experiment is designed by TU Delft and was used as a tugboat simulator. During the project it was also used for these other purposes so it could be used for this project under the condition that no major hardware changes would be made. This section gives a brief overview on how the communication was set up and on the minor changes that were made. The components of which the setup exist are already explained in the scientific paper and are not discussed here again.

3.1. Communication

The Bachmann controller usually only communicates with Matlab, this communication is set up with a third party piece of software and uses a TCP connection. Since the packets used in this communication are unknown it is impossible to directly connect other software to this existing communication line. When connecting other software there are two viable options. The first is to set up a second connection between the Bachmann machine and the operator computer.

One of the downfalls of such a system is that one connection could severely reduce the speed of the other without the other knowing about it. Another problem is that with multiple connections it is hard to know when and from what source changes in the variables can be expected. The last issues is the fact that such a setup would require an additional program to run on the Bachmann controller, this could potentially cause interference with the existing tugboat simulation.

The second possibility is to internally route all necessary communication packets to Matlab which in turn will handle the communication with the Bachmann controller. this is the construction that was used in the project. A block scheme of such a setup can be found in figure 3.1. The internal communication between Matlab and Java was done over a UDP connection with the localhost.

This communication protocol proved to be fast and reliable enough to handle the communication.

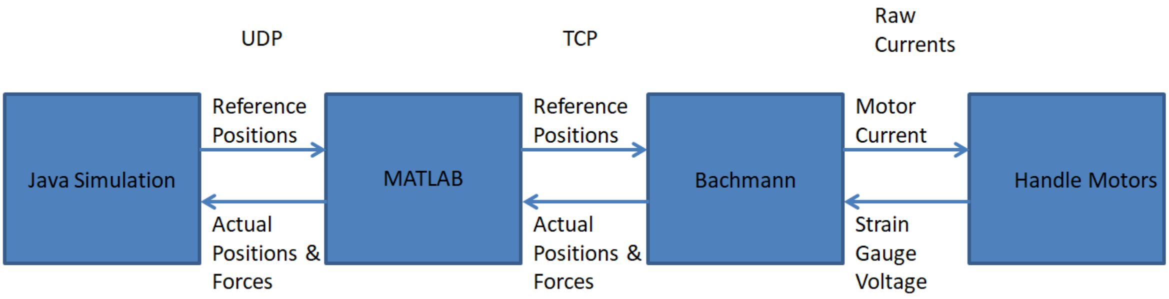


Figure 3.1: A block scheme representation of the communication between the java simulation and the joysticks.

3.2. Choice of Variables

The experiment used force as an input and position of the handle as the feedback variable. Since using one of these variables for both input and feedback is impossible, the only other option was to exchange the variables and use force as the feedback variable and the position of the handle as the input variable. This would have the advantage that the human controller cannot neglect the feedback by overruling the position of the handle. The feedback in this case however would be translated from the orientation of the leg to a force. While with position feedback, the feedback is modality-matched which is the reason that it was chosen over force feedback.

3.3. Simulink

The Bachmann controller runs on a compiled Simulink model, this model takes the reference positions and velocities from the simulation and tracks them using a PD controller. The reference velocities allow for higher gains and a stiffer tuning of the handles. The z-axes rotations are blocked with a zero reference high gain PD controller. The gains of the x and z axis PD-controllers can be found in table 3.1 A motor safety block prevents step inputs and NaN values to be written to the motors and a state machine activates and de-activates the motor amplifiers. The state machine also puts the software in a safe sate when the E-stop is pressed. A full overview of the simulink model can be found in figure 3.2, an overview of the state machine is given in figure 3.3.

Table 3.1: Gains of the PD-control loops in Simulink, found using heuristic tuning.

	K_p $[\frac{Nm}{rad}]$	K_d $[\frac{Nm}{rad\ s^{-1}}]$
X	6.0	0.15
Z	25.0	0.4

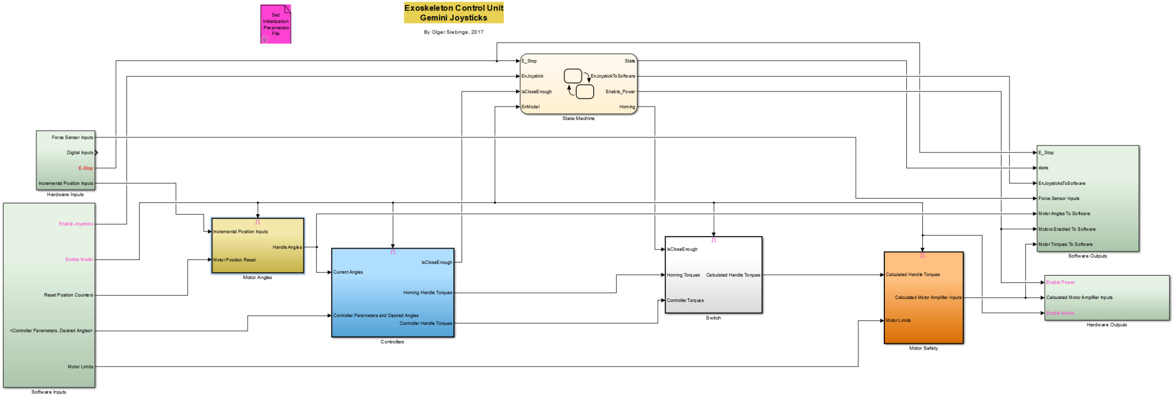


Figure 3.2: A full overview of the Simulink model running on the Bachmann controller, based on the Simulink model used by the tug boat setup. The controller and state machine block are shown in other figures. The motor angles block converts the incremental information to angles and velocities and is not further explained. The motor safety block prevents dangerous motor commands and the switch is a combination of four switches, one for every motor, which switches between the homing and normal controller. The output to the motor amplifiers is clipped to $+-6\ V$ which corrensponds to $I_{max} = 1.8A$ and $\tau^{motor}_{max} = 0.1Nm$. The motor angle and motor safety block are parts of the tugboat Simulink model [1].

The state machine consists of 6 states of which the default state is "disabled". The software will be in this state if another model is running on the Bachmann machine. In this state all outputs of the exoskeleton model are disabled. If the model is enabled, meaning the tug boat simulation is disabled, the software will start in the stopped state. If the enable joysticks Boolean is flipped it will transition to the initializing state. This state was included in the design with no specific purpose but for future use of initialization procedures. However there was never a need for any initialization so this state is unused and the software will directly transition to the homing state.

In the homing state the motor power is enabled and the joysticks will move close to their reference position using a velocity controller ($\dot{q}_{ref} = 0.5\ rad\ s^{-1}$, $k^x_d = 0.3$, $k^z_d = 0.6$). The homing controller block is shown in figure 3.4. A velocity controller was used to guarantee that the joystick will move steadily to their initial

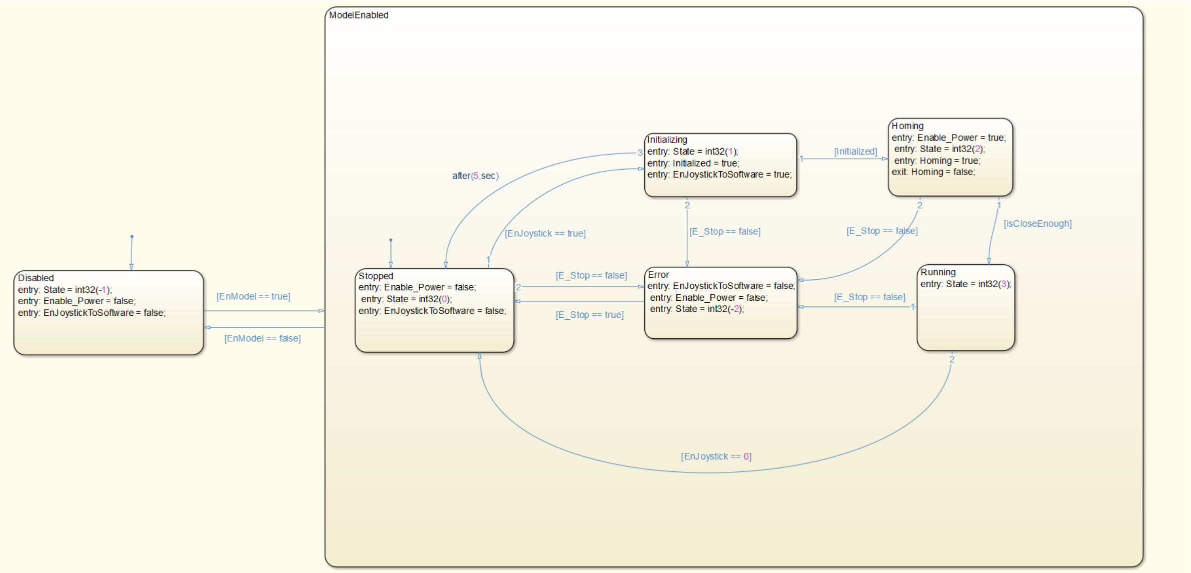


Figure 3.3: An overview of the state machine in the simulink model. This state machine runs on the Bachmann controller. It consists of 6 states which are explained in this section.

position regardless of their current position. It also ensures that the force from the motor is constant in case the joystick is blocked, there is no increasing position error. This makes it a safer alternative than a moving reference position controller.

When the joysticks are close enough ($< 0.05\ rad$) to the current reference position the state machine transitions to the running state. In this state the controller switches to the regular PD controller that uses the reference position and velocity from the simulation. A combination of reference position and velocity is used to increase the stability of the controller and allow for higher gains and better tracking. An overview of the regular controller can be found in figure 3.5.

If the E-stop is pressed during any of these states it will not only disable the motor power hardware wise, it will also put the system in error state. Disabling all outputs and making sure the system will start in the stopped state again when the E-stop is released.

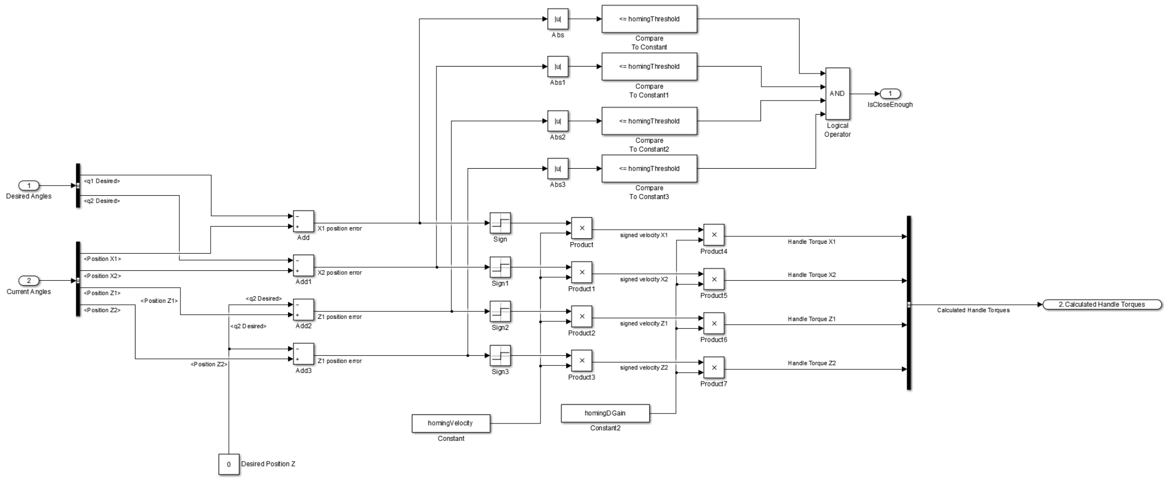


Figure 3.4: An overview of the homing controller block of the simulink model. This is a velocity controller using $\dot{q}_{ref} = 0.5\ rad\ s^{-1}$, $k_d^x = 0.3$, $k_d^z = 0.6$ with a homingThreshold of 0.05 rad

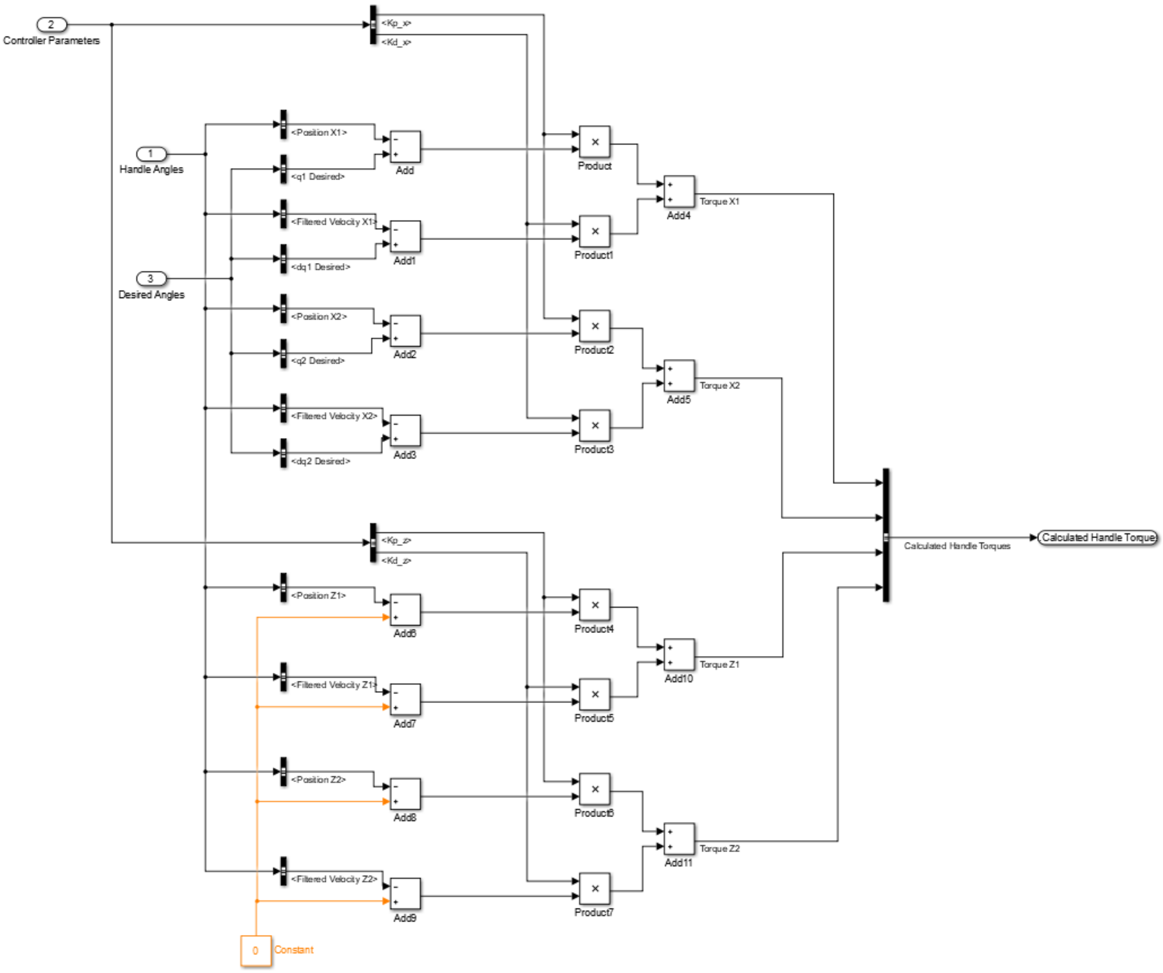


Figure 3.5: An overview of the regular motor controller from the simulink model. It uses a 0 reference on the z-axes motors and a reference from the simulation for the x-axis motors. The used gains can be found in table 3.1.

3.4. Strain Gauges

The only major adjustment made to the setup for this project was the connection of the strain gauges. These strain gauges where pre-fitted to the joysticks and connected in a full Wheatstone bridge setup. However they where directly connected to the Bachmann controller which is not capable of delivering the needed 10 V excitation voltage with low enough noise to get usable measurements. A second issue was that the strain gauges were placed on a part of the handle with a high stiffness. This resulted in the measuring of only very minor values, with the 10 V excitation voltage the maximum measured difference was in the order of 0.3 mV. The Bachmann controller was not capable of measuring these tiny differences on its analog input ports.

These problems were solved by using separate sensor amplifiers. These amplifiers delivered a very clean excitation voltage and amplified the measured signal 1000 times. This resulted in a clean signal that could be perceived by the Bachmann controller.

4 Experiment Design

The design of the experiment was an iterative process which can roughly be divided in three phases. The first step was a pre-pilot this was done by myself and consisted of doing the whole experiment to try and identify potential problems. After this pre-pilot a pilot was done with two participants that had not seen the setup or software before. Finally there was a second pilot with only one participant to check if some of the changes to the experiment design had the desired effect. These three phases are discussed in this chapter.

4.1. Pre-Pilot

Before the pre-pilot some runs without human input were executed to investigate the behaviour of the disturbance generator. The first version of the disturbance generator used a random direction for every disturbance. It also used normally distributed time intervals between the disturbances. This resulted in a very big variance in the measured data because in a "lucky" run the disturbances counter acted each other because their direction changed at every push. Also it could happen that most of the disturbances happened at a stable moment, for instance during double support or just before heel strike. In these runs the exoskeleton would walk stable for a long time.

But there were also runs where a number of disturbance with the same direction occurred in a row, at the most unstable point of a step. During these runs the exoskeleton would fall quickly. To limit the variance in the comparison data a new version of the disturbance generator was designed. This version works as explained in the paper, it chooses one direction for the entire run and applies the disturbance at the same time during swing. The interval is now a normally distributed step interval. In contrary to the explanation in the paper, at this point the disturbance generator used the same offset and force increase for forwards and backwards disturbances.

After the comparison data collection, the experiment was executed by myself under four different conditions. The first was with full feedback as described in the scientific paper. Under the second condition the position feedback was disabled. And both of these conditions where repeated with limited visual feedback. This was achieved by lowering the frame rate of the simulation. The different conditions were meant to investigate the role of the mechanical and visual feedback.

The set with full feedback was done twice, once with swing time and once with step length control. After this first check the decision was made to use swing time since this resulted in higher scores. The pre-pilot showed that is was possible to record 6 runs under each condition in just under two hours while leaving some time for training runs. However it also became clear that short breaks were needed after every five runs to prevent fatigue. Nothing else was altered and the first pilot was done.

4.2. Pilot I

In the first pilot two participants executed the experiment under the four different conditions. It became clear pretty quickly that the number of conditions was to high for the available training time. Participants were confused and not ready for the real measurements after the training period. This could also be seen in the results. Under the full feedback the participants seemed to score better than in the case without human input but under the other conditions there was no clear difference. This resulted in a radical change in the experiment design.

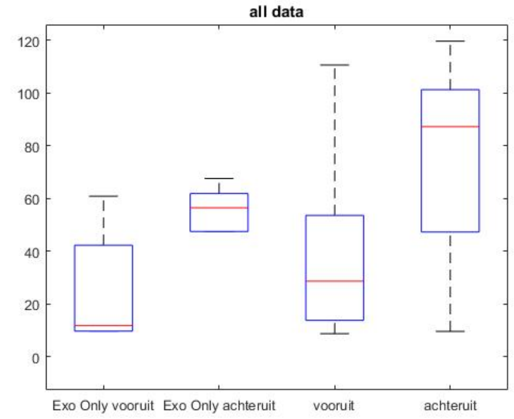


Figure 4.1: The results of the exo only and the participant of the second pilot study grouped on disturbance direction (forwards, vooruit and backwards, achteruit). The y-axis shows the number of successfully completed steps.

It turned out that the available time per participant was simply not enough to test all conditions. Dividing the participants in groups and let every group work under a different condition would require to many participants for the scale of this project. So the decision was made to focus on proofing the feasibility of the control method and not investigate the effects of the different kinds of feedback. In the new design all runs would be executed under full feedback conditions.

There were signs that the participants performed better in one of the two perturbation directions but since the data was so fragmented due to the four conditions it was hard to draw any conclusions on that at this point.

4.3. Pilot II

Because of this radical change a second pilot study was performed with only one participant. This participant completed the experiment as described in the scientific paper with the swing time control parameter. The results of this pilot were analyzed and tested for statistical differences against the no human runs. Since the data was not normally distributed and the distributions of the groups were different a test based on mean ranks had to be used. The original experiment design used the time until the fall of the exoskeleton as the measured variable. But in combination with the mean rank test it proofed difficult to get a significant result. For this reason the number of successfully completed steps was used which is an ordinal variable and thus more suited for a mean rank test.

With more data points under one condition it became possible to investigate the potential effect of push direction on the results. When the results of the participant and the exoskeleton only were grouped on disturbance direction the plot in figure 4.1 was obtained. This plot shows that not only the participant performed different in both situations, but also when there was no human in control the exoskeleton behaved differently. This indicated a problem with the disturbance generator.

To investigate this problem histograms of the fall time with disturbance forwards and backwards were created, these histograms can be found in figure 4.2. These histograms show clear differences in the distributions. If these differences are minimized all data can be treated the same. So the offset and increments of the backwards directed force were tuned until the histogram in figure 4.3 could be obtained. This new disturbance generator was used in the final experiment, its parameters can be found in the paper.

The final data showed significant results in the forward perturbation direction. This indicated that the significant results could be reached with a small group of participants. So at this point it was decided that the group would be big enough to be split up in two subgroups. This way it became possible to investigate the effects of the different control parameter (swing time vs step length).

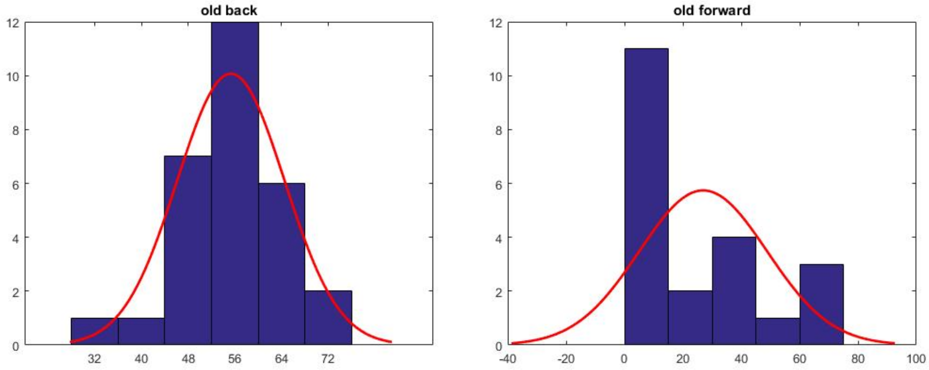


Figure 4.2: Histograms of the fall times with forward and backward perturbations without human control with the old disturbance generator. The red lines are matlab's attempt to fit a normal distribution and are not very relevant.

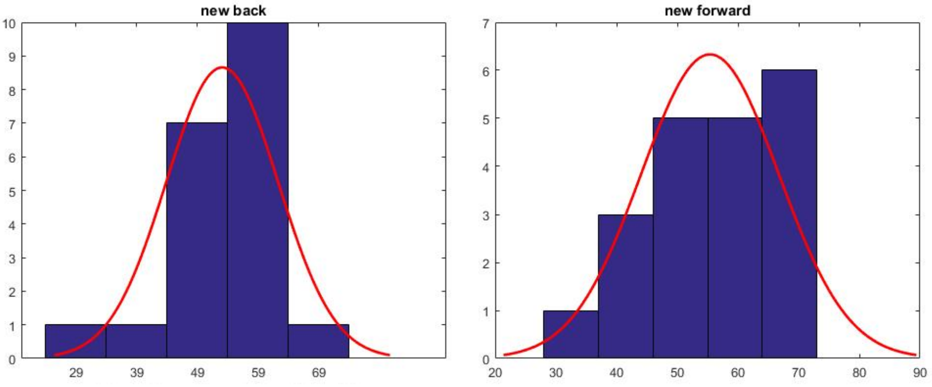
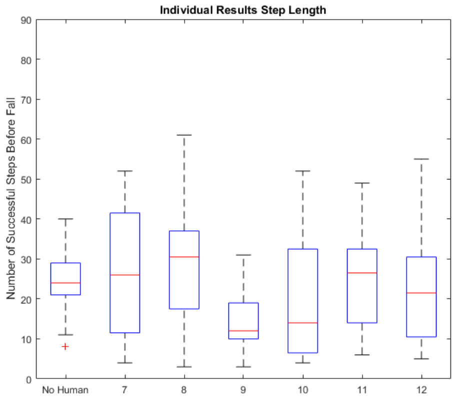


Figure 4.3: Histograms of the fall times with forward and backward perturbations without human control with the new disturbance generator. The red lines are matlab's attempt to fit a normal distribution and are not very relevant.

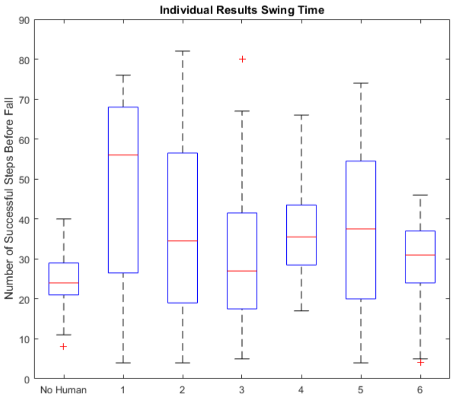
5

Raw Data

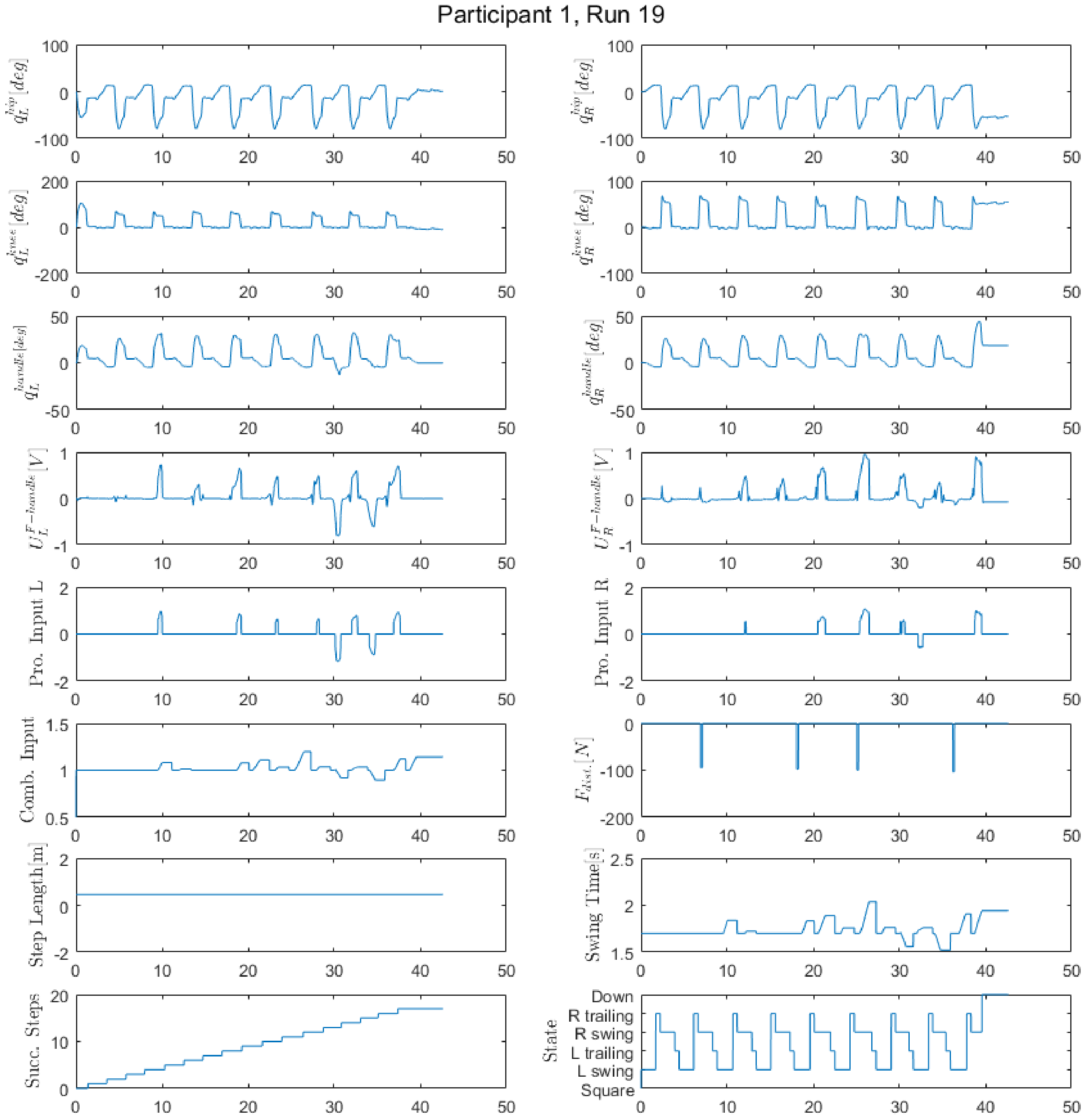
The exoskeleton controller and simulation environment use a total of 644 YoVariables which are all stored as time traces after every run. The final values of a run of all these variables where separately stored in a *.csv file for easy access to the successful number of steps variable. Twelve participants each completed 20 runs which were marked for data collection and another 11 training runs. This gives a total of $12 * 31 = 372$ data sets with $372 * 644 = 239.568$ variable time traces between them. It would be impossible to provide all this raw data here. But to give insight in the data and show the differences between the subjects, a selected number of variables of run 19 is given here for every participant. The individual performances of the participants are also provided.



(a) The individual performance results of the step length participants.

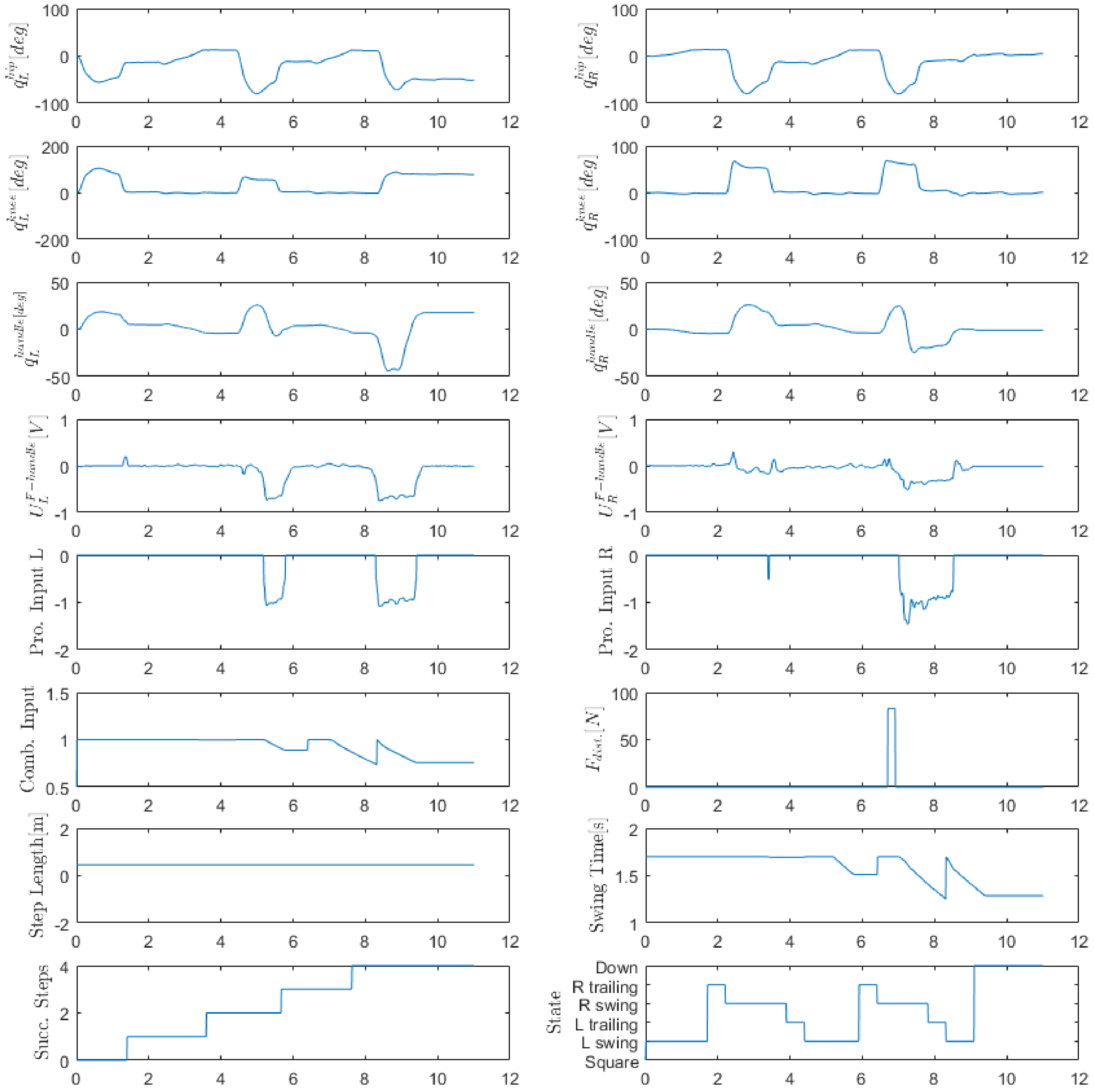


(b) The individual performance results of the swing time participants.

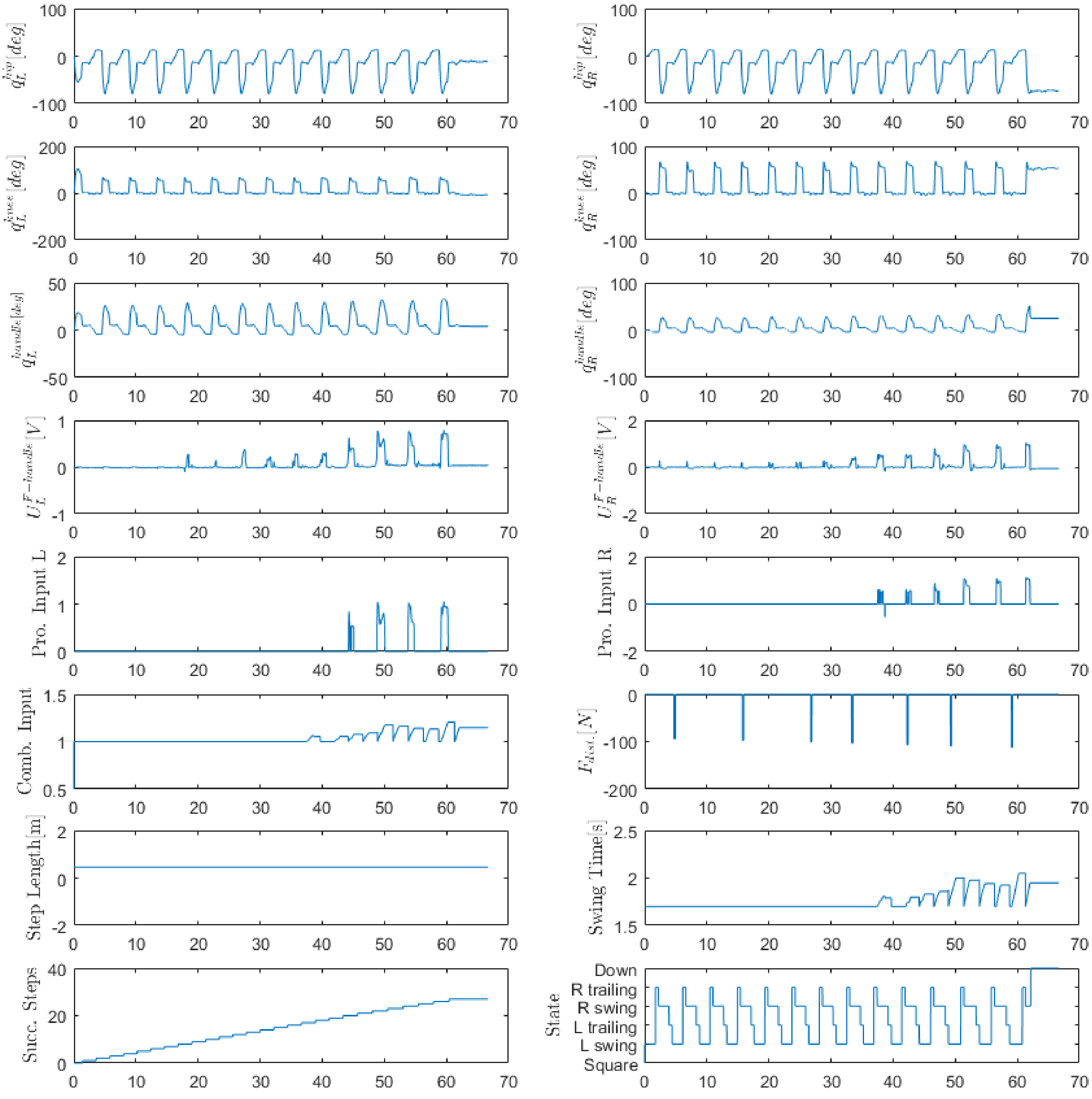


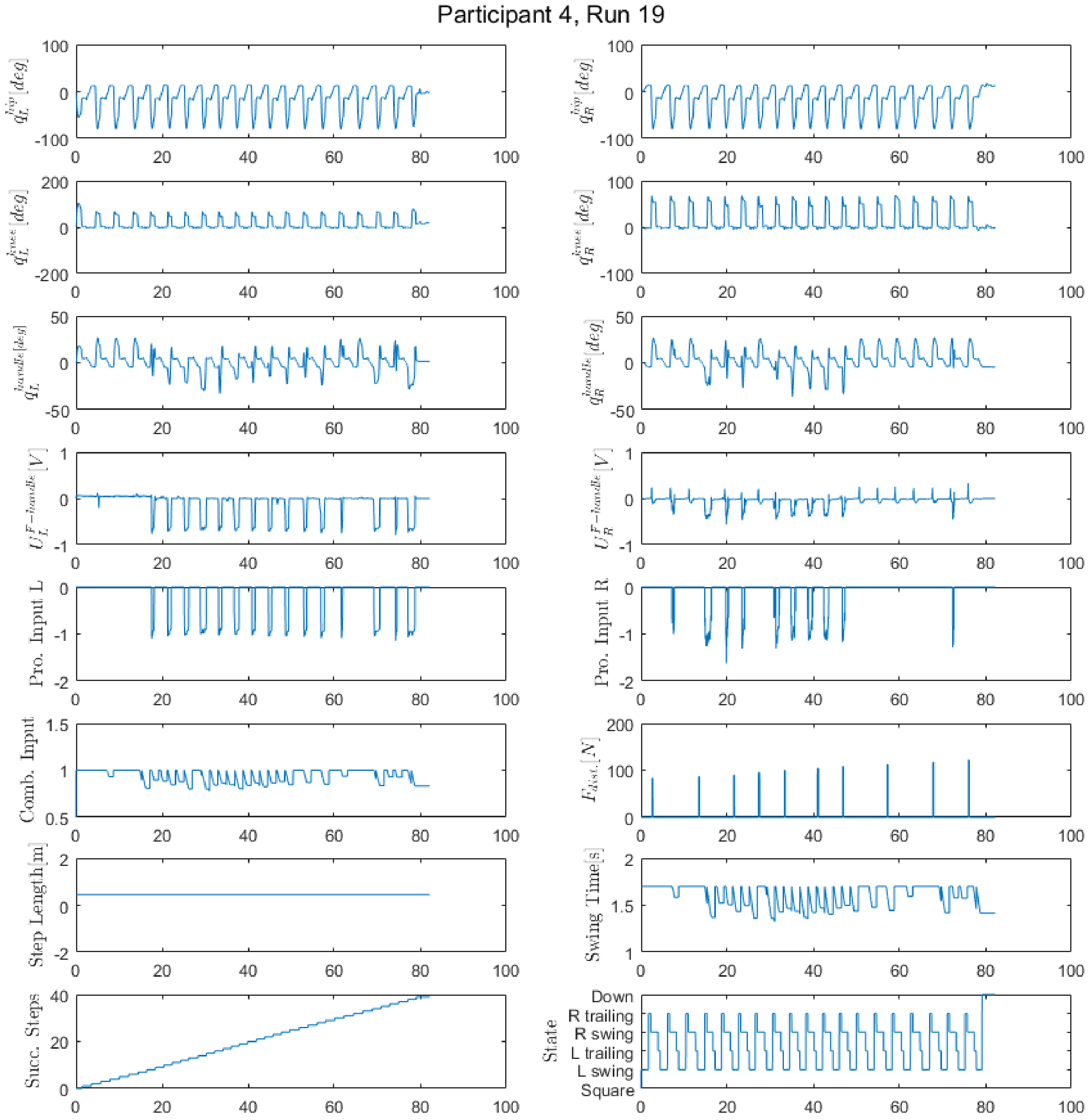
Participant 1, Run 19
$q_L^{hip}\ [deg]$
100
0
-100
$q_R^{hip}\ [deg]$
100
0
-100
$q_L^{knee}\ [deg]$
200
0
-200
$q_R^{knee}\ [deg]$
100
0
-100
$q_L^{handle}[deg]$
50
0
-50
$q_R^{handle}\ [deg]$
50
0
-50
$U_L^{F-handle}\ [V]$
1
0
-1
$U_R^{F-handle}\ [V]$
1
0
-1
Pro. Input L
2
0
-2
Pro. Input R
2
0
-2
Comb. Input
1.5
1
0.5
$F_{dist.}\ [N]$
0
-100
-200
Step Length[m]
2
0
-2
Swing Time[s]
2.5
2
1.5
Succ. Steps
20
10
0
State
Down
R trailing
R swing
L trailing
L swing
Square
0
10
20
30
40
50

Participant 2, Run 19

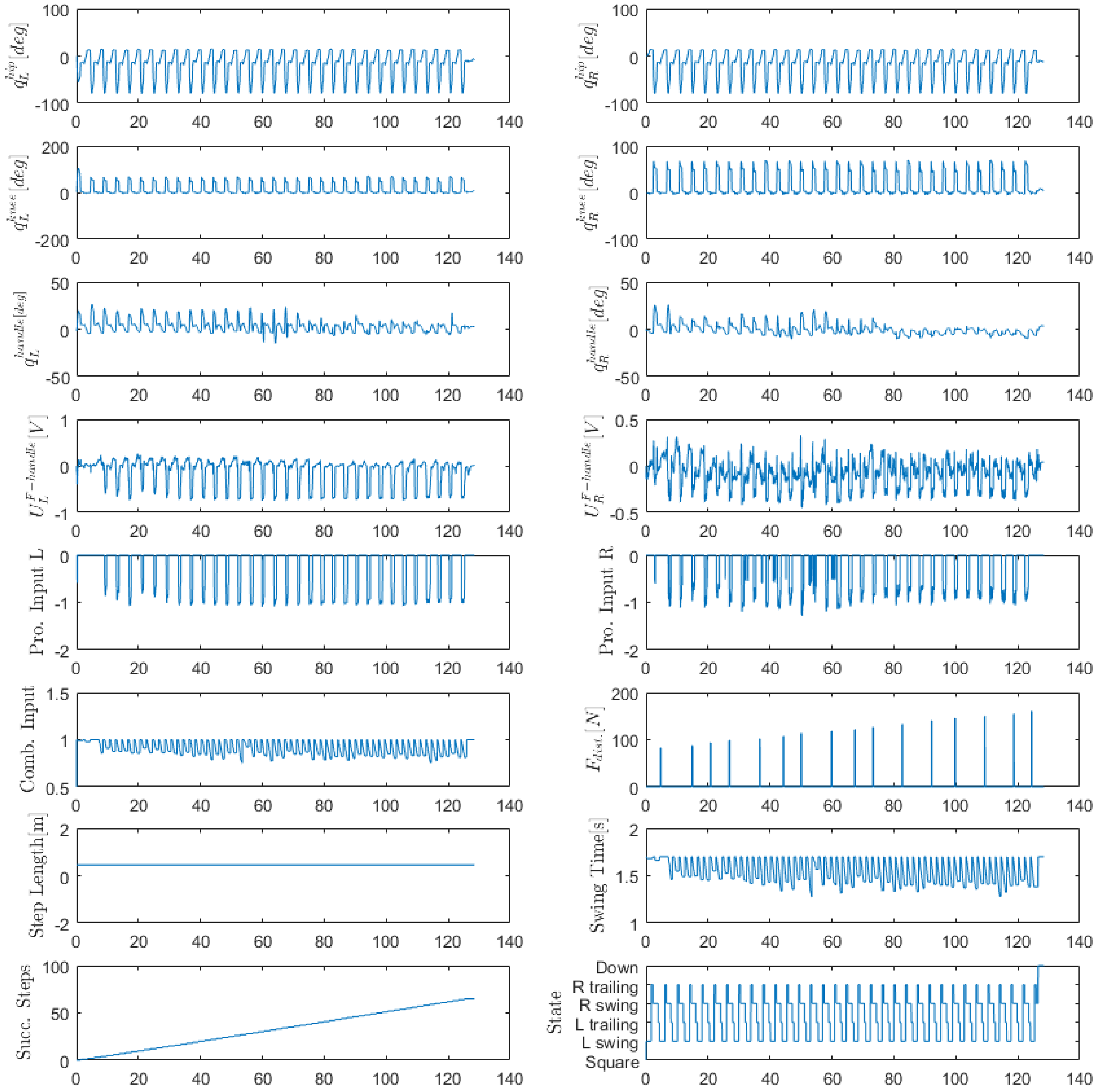


Participant 3, Run 19

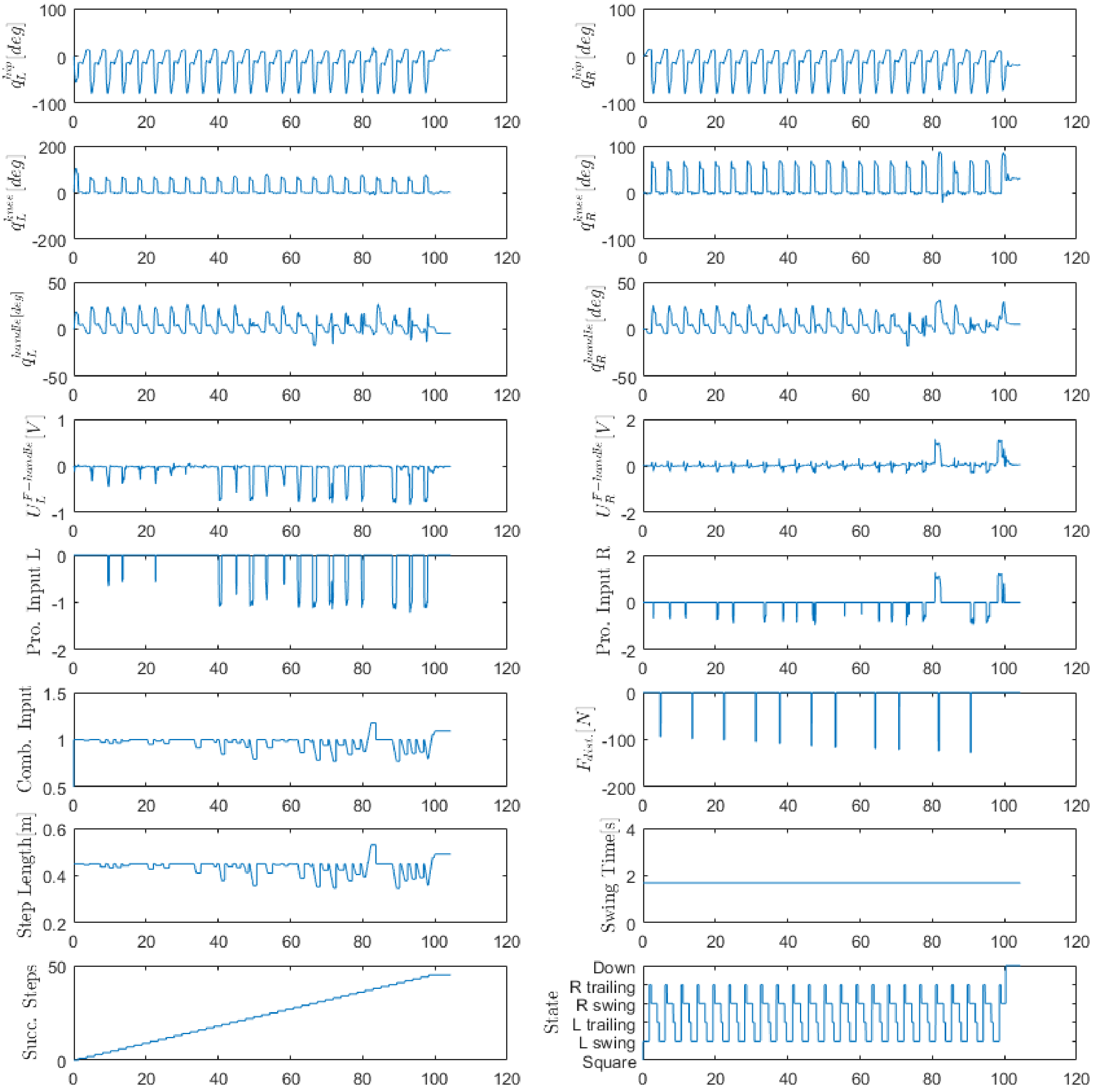


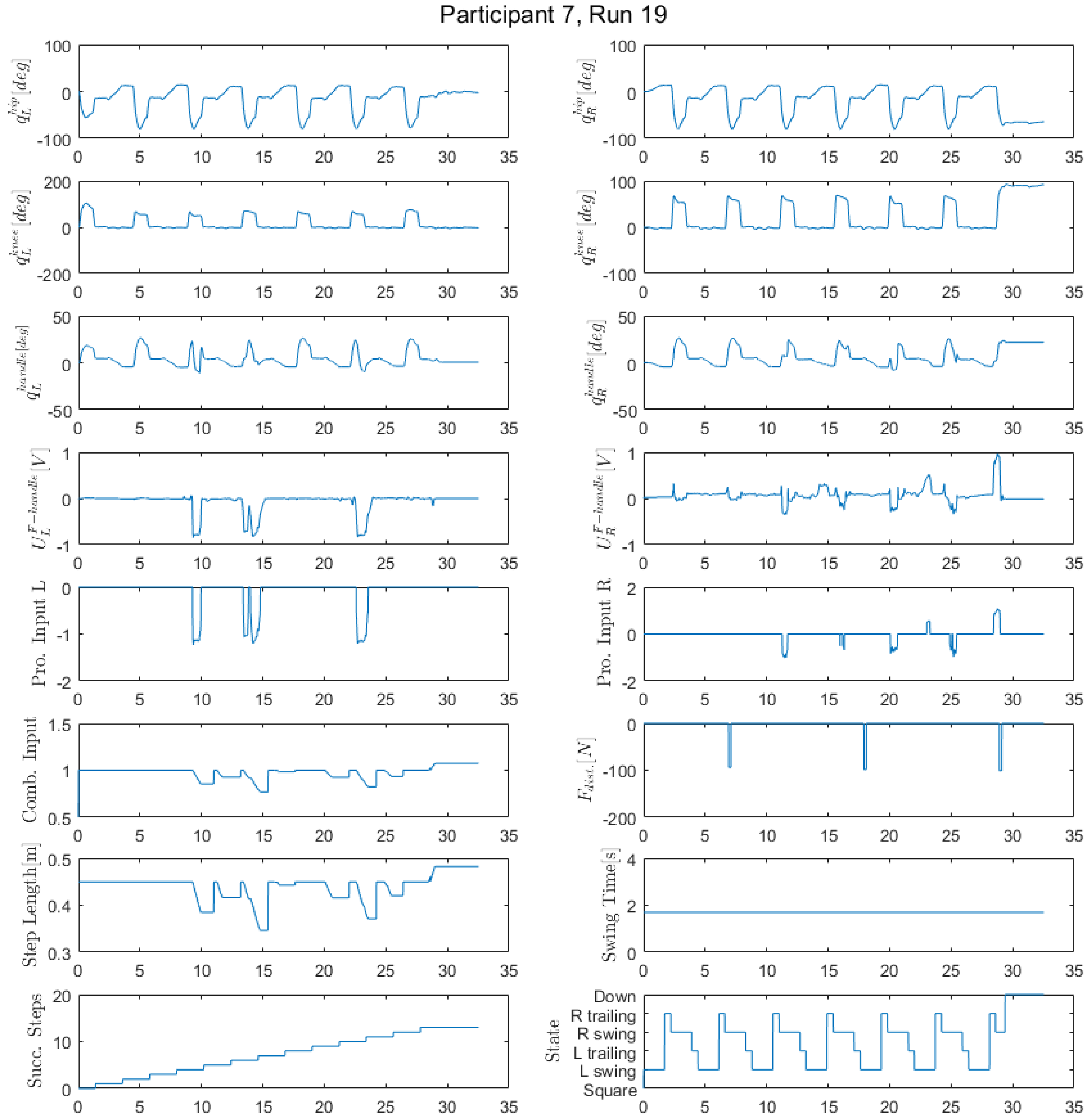

Participant 4, Run 19
q_L^hip [deg]
q_R^hip [deg]
100
0
-100
q_L^knee [deg]
200
0
-200
q_R^knee [deg]
100
0
-100
q_L^handle [deg]
q_R^handle [deg]
50
0
-50
U_L^F-handle [V]
U_R^F-handle [V]
1
0
-1
Pro. Input L
Pro. Input R
0
-1
-2
Comb. Input
1.5
1
0.5
F_dist. [N]
200
100
0
Step Length[m]
2
0
-2
Swing Time[s]
2
1.5
1
Succ. Steps
40
20
0
State
Down
R trailing
R swing
L trailing
L swing
Square
0
20
40
60
80
100

Participant 5, Run 19

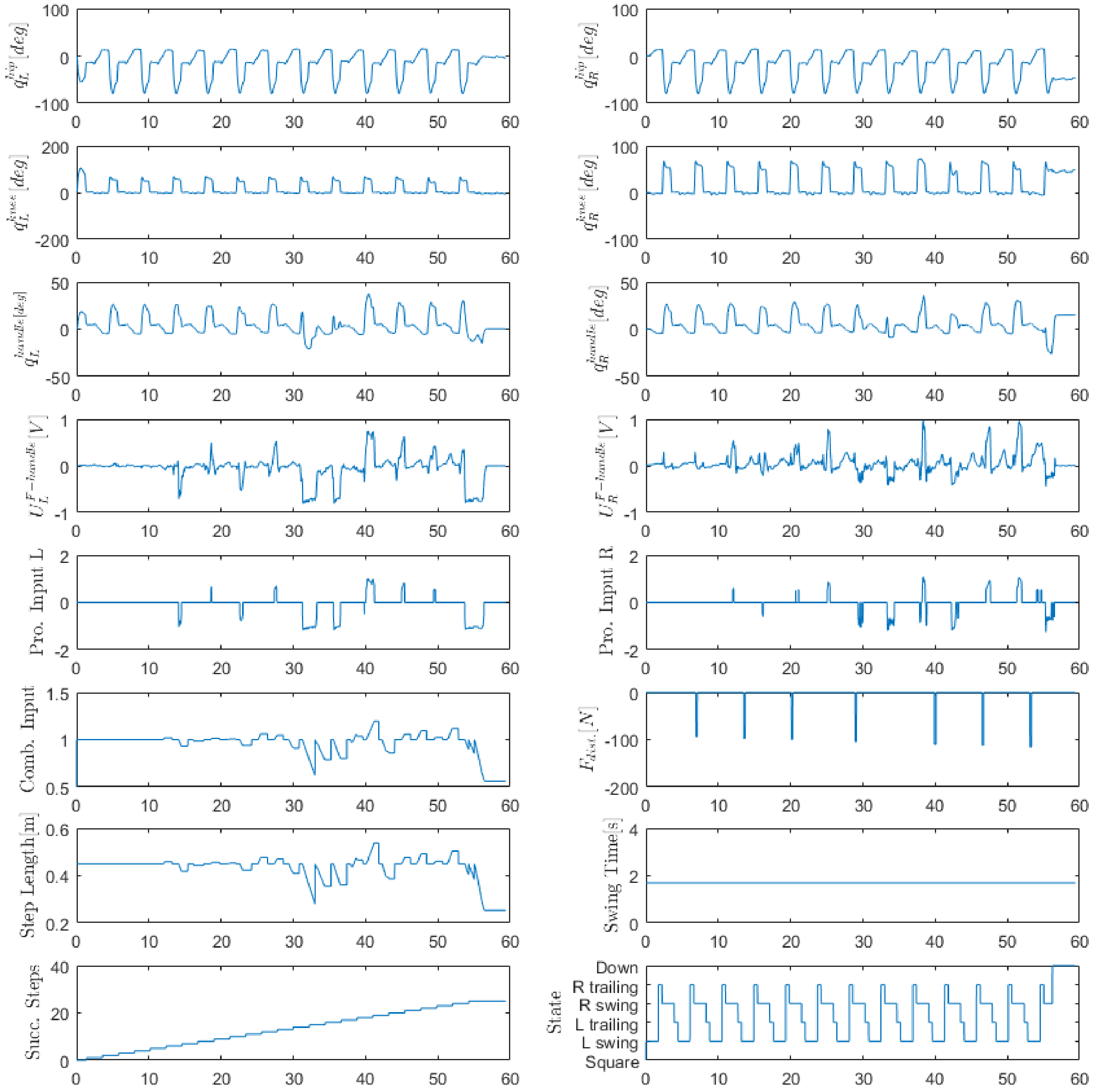


Participant 6, Run 19

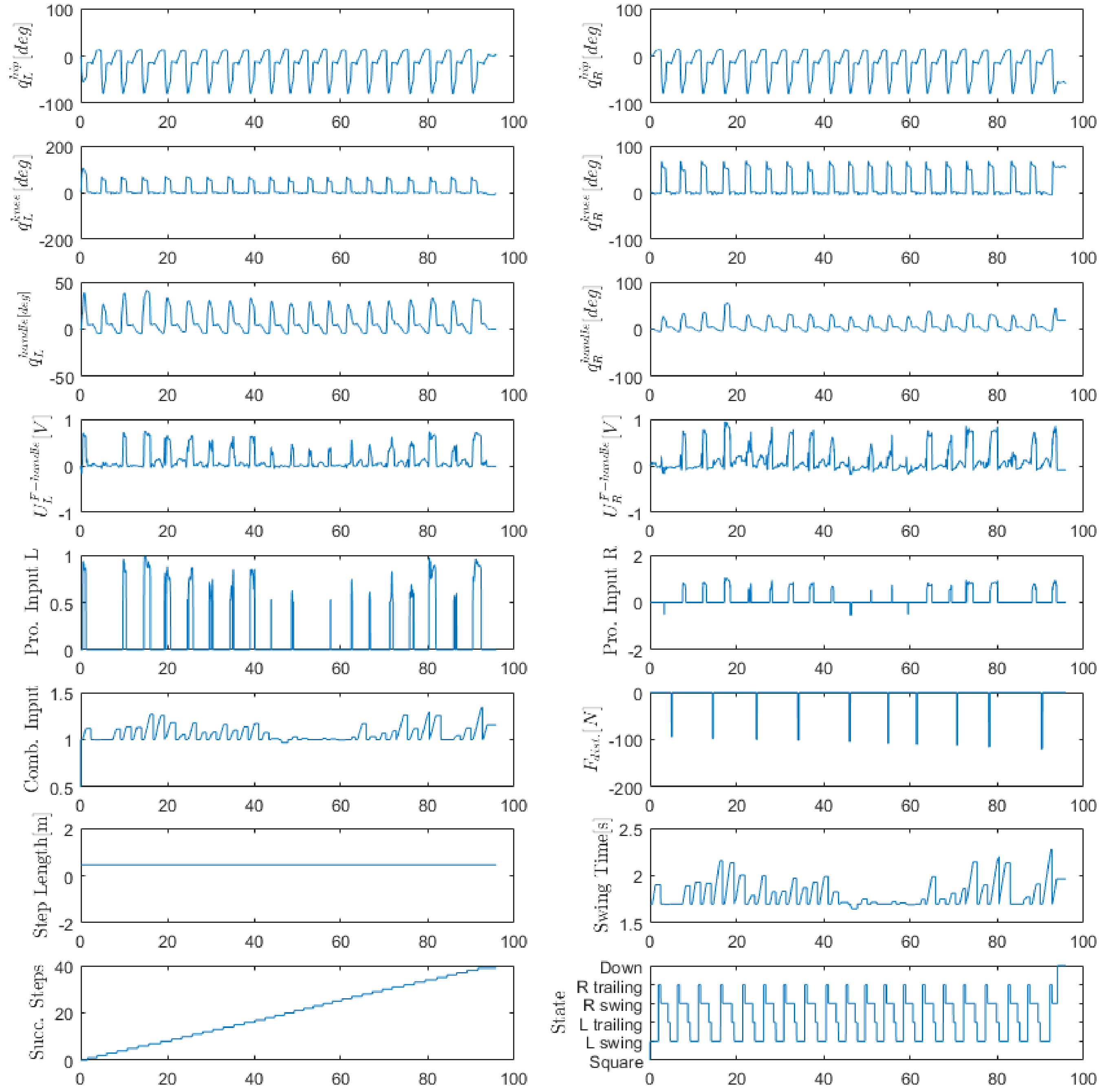



Participant 7, Run 19
$q_L^{hip}\,[deg]$
$q_R^{hip}\,[deg]$
$q_L^{knee}\,[deg]$
$q_R^{knee}\,[deg]$
$q_L^{handle}\,[deg]$
$q_R^{handle}\,[deg]$
$U_L^{F-handle}\,[V]$
$U_R^{F-handle}\,[V]$
Pro. Input L
Pro. Input R
Comb. Input
$F_{dist.}\,[N]$
Step Length[m]
Swing Time[s]
Succ. Steps
State
Down
R trailing
R swing
L trailing
L swing
Square
100
0
-100
200
-200
50
-50
1
-1
2
-2
1.5
0.5
0.5
0.4
0.3
4
20
10
0
5
10
15
20
25
30
35

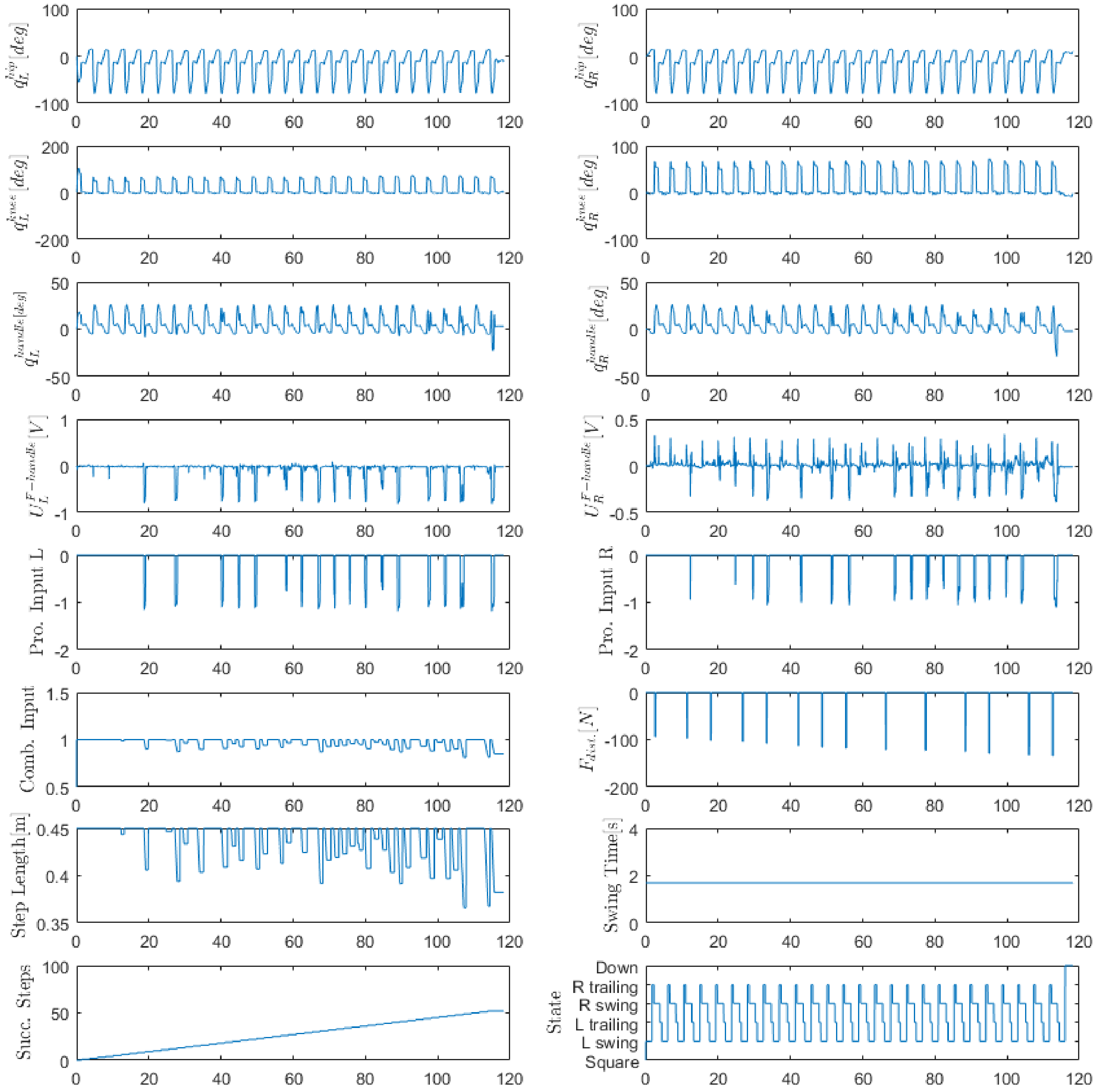
Participant 8, Run 19



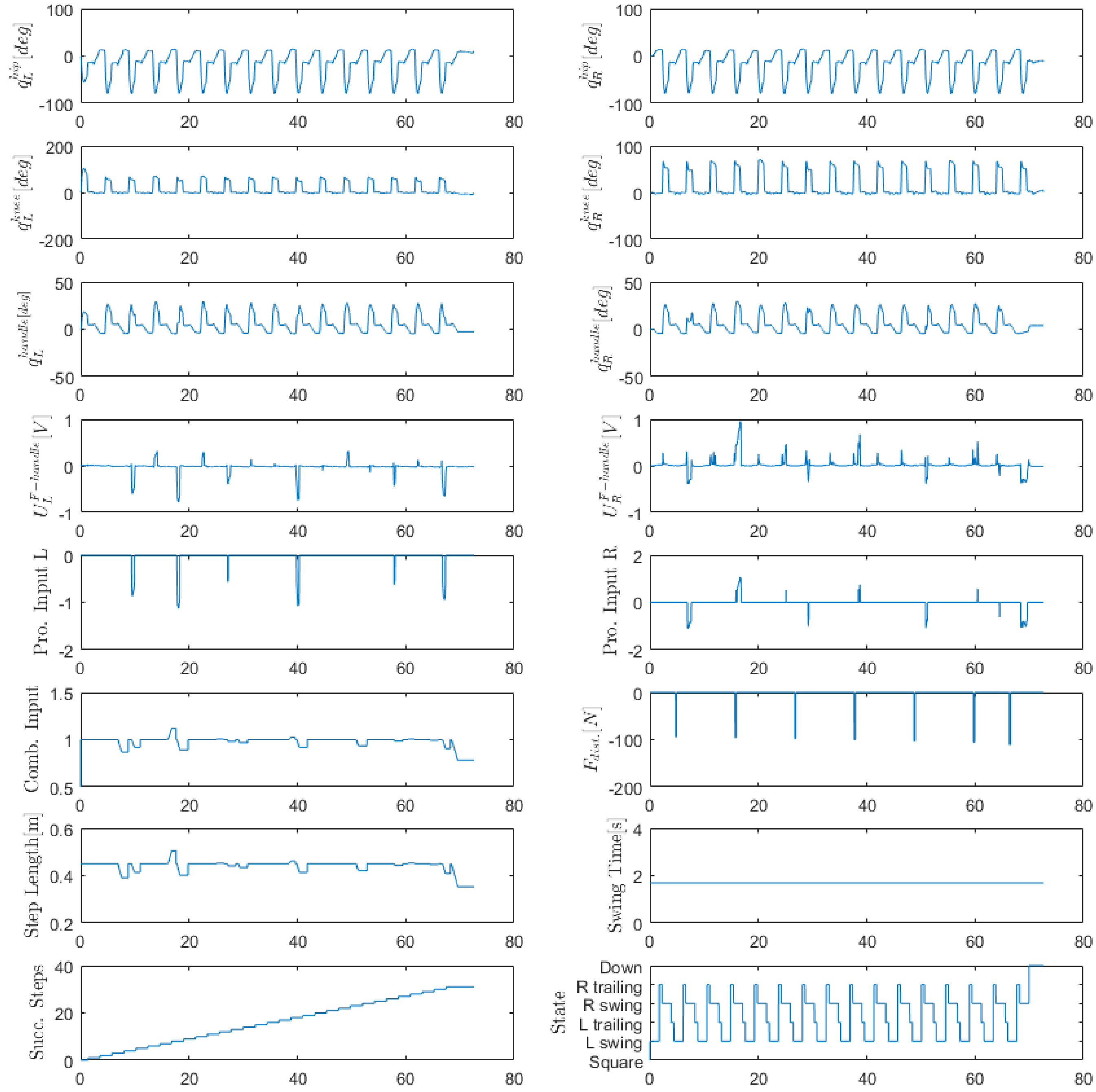
Participant 9, Run 19



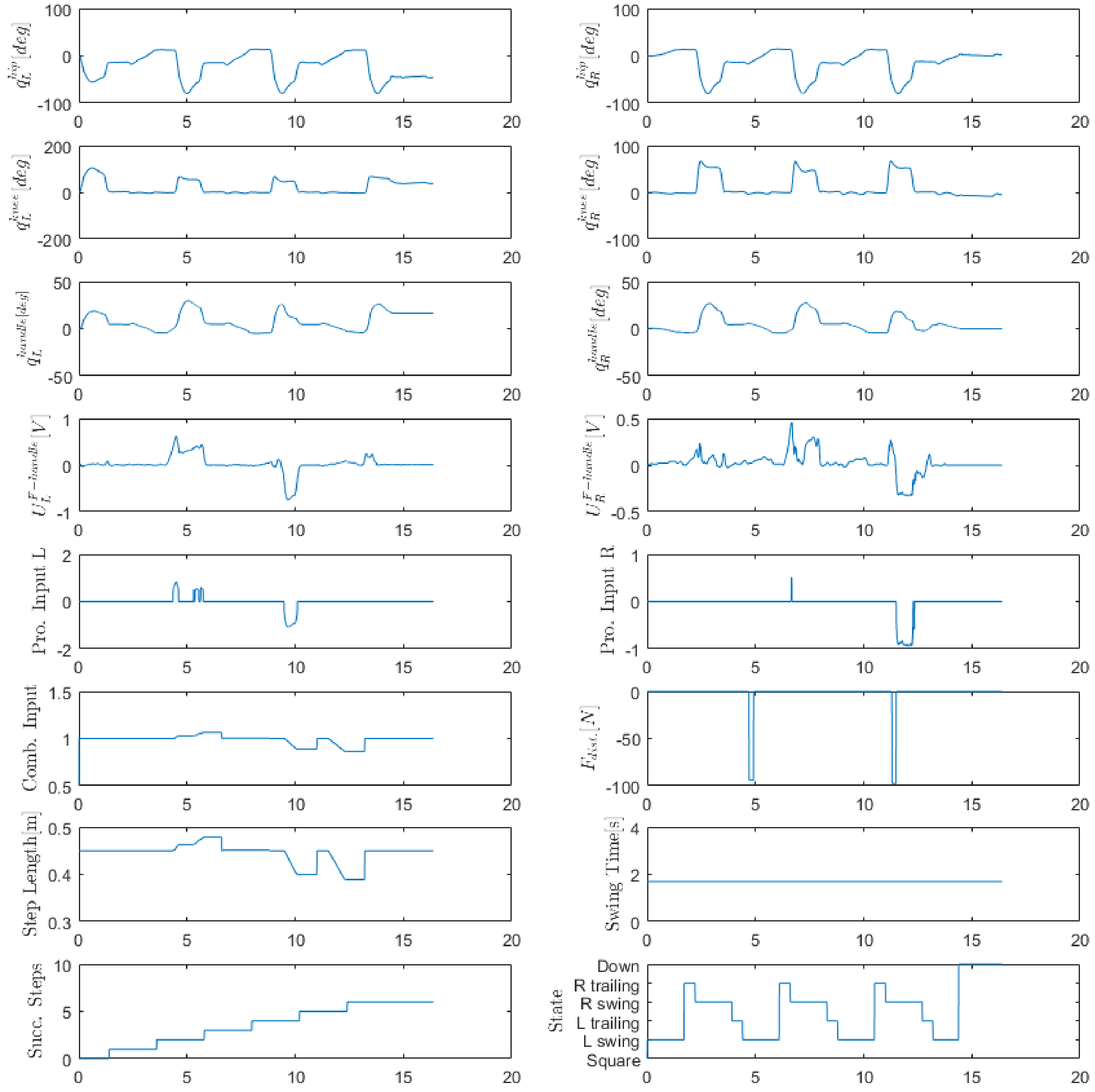
Participant 10, Run 19



Participant 11, Run 19



Participant 12, Run 19



6

Ethics Form

Delft University of Technology
ETHICS REVIEW CHECKLIST FOR HUMAN RESEARCH

This checklist should be completed for every research study that involves human participants and should be submitted before potential participants are approached to take part in your research study.

In this checklist we will ask for additional information if need be. Please attach this as an Annex to the application.

Please upload the documents (go to this page for instructions).

Thank you and please check our website for guidelines, forms, best practices, meeting dates of the HREC, etc.

I. Basic Data

Project title:	**Human Machine Interface for Exoskeleton**
Name(s) of researcher(s):	**O. Siebinga**
Research period (planning)	**Half October – half November 2017**
E-mail contact person	
Faculty/Dept.	**3me/CoR**
Position researcher(s):[1]	**Master Student**
Name of supervisor (if applicable):	**Dr. Ir. D.A. Abbink**
Role of supervisor (if applicable):	**Associate professor**

II. A) Summary Research

The goal of this research is to determine whether humans are capable of correcting for disturbances when walking in an exoskeleton for paraplegics. In this study a haptic hand-interface will be evaluated, which should allow for correction/adaptation of the exoskeleton gait using the hands.

The participants (12 able bodied subjects) will use two existing haptic joysticks (http://www.delfthapticslab.nl/device/haptic-control-units-for-maritime-applications) to control a computer simulation of a walking exoskeleton. The positions of the joysticks provide feedback on the position of the legs of the exoskeleton and by exerting forces on the joysticks, the gait can be changed. As an extra variable the visual feedback from the simulation will be confined. The effect of this limited visual feedback on the usefulness of the position feedback will be tested.

B) Risk assessment

The only potential risk is associated with the hardware setup and can be considered as twofold, mechanical and electrical. The servo motors in the setup exert force on the handles which are held by the subject. However these forces are limited by the stall torque of the motors (F_Z = 192 N & F_X = 102 N) and the software (F_Z = 18.2 N & F_X = 9.7 N). Since the software limits are the lowest, these represent the maximum forces acting on the participant. Such low forces are unlikely to cause pain or harm. An emergency stop connected to the system can be used to put the software in a safe state and disable all motor power through a relay.

The setup is connected to the main power grid but the electrical system of the setup is isolated from the handles used by the participant. All conduction parts of the setup are connected to the ground. And all sub-systems (e.g. sensors, motors) have separate fuses.

[1] For example: student, PhD, post-doc

III. Checklist

	Yes	No
1. Does the study involve participants who are particularly vulnerable or unable to give informed consent? (e.g., children, people with learning difficulties, patients, people receiving counselling, people living in care or nursing homes, people recruited through self-help groups).	□	■
2. Are the participants, outside the context of the research, in a dependent or subordinate position to the investigator (such as own children or own students)?[2]	□	■
3. Will it be necessary for participants to take part in the study without their knowledge and consent at the time? (e.g., covert observation of people in non-public places).	□	■
4. Will the study involve actively deceiving the participants? (e.g., will participants be deliberately falsely informed, will information be withheld from them or will they be misled in such a way that they are likely to object or show unease when debriefed about the study).	□	■
5. Will the study involve discussion or collection of information on sensitive topics? (e.g., sexual activity, drug use, mental health).	□	■
6. Will drugs, placebos, or other substances (e.g., drinks, foods, food or drink constituents, dietary supplements) be administered to the study participants?	□	■
7. Will blood or tissue samples be obtained from participants?	□	■
8. Is pain or more than mild discomfort likely to result from the study?	□	■
9. Does the study risk causing psychological stress or anxiety or other harm or negative consequences beyond that normally encountered by the participants in their life outside research?	□	■
10. Will financial inducement (other than reasonable expenses and compensation for time) be offered to participants?	□	■

Important:
if your answered 'yes' to any of the questions mentioned above, please submit the full application form to HREC
(see: HREC website for forms or examples).

[2] **Important note concerning questions 1 and 2.** Some intended studies involve research subjects who are particularly vulnerable or unable to give informed consent .Research involving participants who are in a dependent or unequal relationship with the researcher or research supervisor (e.g., the researcher's or research supervisor's students or staff) may also be regarded as a vulnerable group . If your study involves such participants, it is essential that you safeguard against possible adverse consequences of this situation (e.g., allowing a student's failure to complete their participation to your satisfaction to affect your evaluation of their coursework). This can be achieved by ensuring that participants remain anonymous to the individuals concerned (e.g., you do not seek names of students taking part in your study). If such safeguards are in place, or the research does not involve other potentially vulnerable groups or individuals unable to give informed consent, it is appropriate to check the NO box for questions 1 and 2. Please describe corresponding safeguards in the summary field.

	Yes	No
11. Will the experiment collect and store videos, pictures, or other identifiable data of human subjects? [3]	□	■
If "yes", are you sure you follow all requirements of the applicable data protection legislation? *(Please provide proof by sending us a copy of the informed consent form).*	□	□
12. Will the experiment involve the use of devices that are not 'CE' certified?	■	□
Only If 'yes': continue with the following questions:		
• Was the device built in-house?	■	□
• Was it inspected by a safety expert at TU Delft? (*Please provide device report, see: HREC website*)	■	□
• If it was not built in house and not CE-certified, was it inspected by some other, qualified authority in safety and approved? *(Please provide records of the inspection).*	□	□
13. Has or will this research be submitted to a research ethics committee other than this one? (*if so, please provide details and a scan of the approval or submission if available).*	□	■

IV. Enclosures (tick if applicable)

○ Full proposal (if 'yes' to any of the questions 1 until 10)
○ Informed consent form (if 'yes' to question 11)
✓ Device report (if 'yes' to question 12)
○ Approval other HREC-committee (if 'yes' to question 13)
○ Any other information which might be relevant for decision making by HREC

V. Signature(s

Signature(s) of researcher(s)
Date:

Signature research supervisor (if applicable)
Date:

[3] Note: you have to ensure that collected data is safeguarded physically and will not be accessible to anyone outside the study. Furthermore, the data has to be de-identified if possible and has to be destroyed after a scientifically appropriate period of time.

Delft University of Technology
INSPECTION REPORT FOR DEVICES TO BE USED IN CONNECTION WITH HUMAN SUBJECT RESEARCH

This report should be completed for every experimental device that is to be used in interaction with humans and that is not CE certified or used in a setting where the CE certification no longer applies[1]. The first part of the report has to be completed by the researcher and/or a responsible technician. Then, the safety officer (AMA – Arbo en milieu adviseur) of the corresponding faculty has to inspect the device and fill in the second part of this form. Please visit https://intranet.tudelft.nl/arbeidsomstandigheden/arbeidsomstandigheden/overzicht-amas/ for more information.

Note that in addition to this, all experiments that involve human subjects have to be approved by the Human Research Ethics Committee of TU Delft. You can find more information on the procedures at http://www.hrec.tudelft.nl/

Device identification (name, location): haptic-control-units-for-maritime-applications, Maritime Lab, Building part B

Configurations inspected[2]: NA

Type of experiment to be carried out on the device:[3] Control of a simulation

Name(s) of applicants(s):

Job title(s) of applicants(s):

(Please note that the inspection report should be filled in by a TU Delft employee. In case of a BSc/MSc thesis project, the responsible supervisor has to fill in and sign the inspection report.)

Date:

Signature(s):

[1] Modified, altered, used for a purpose not reasonably foreseen in the CE certification

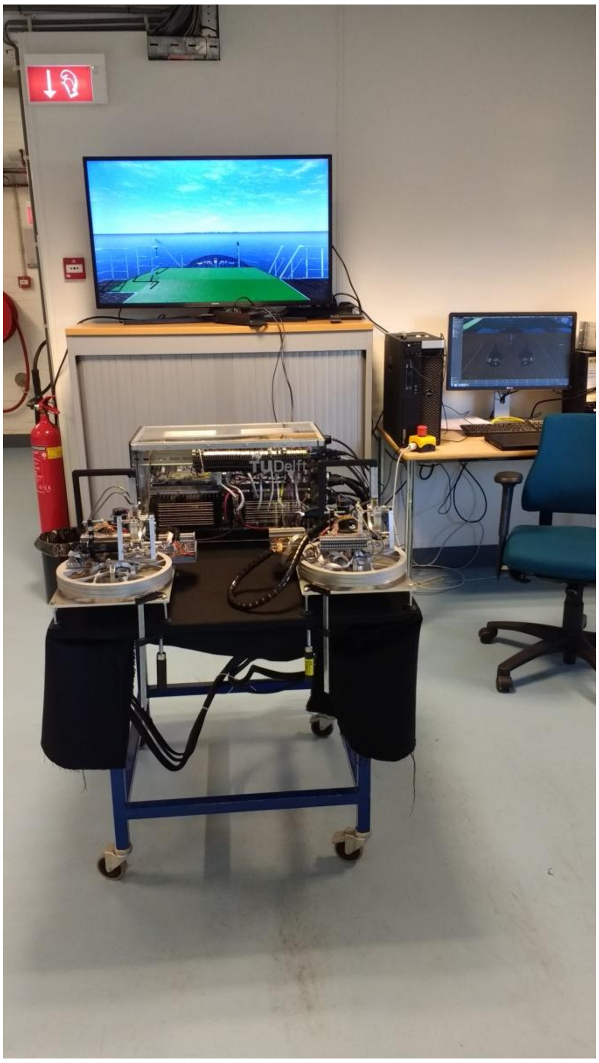
[2] If the devices can be used in multiple configurations, otherwise insert NA

[3] e.g. driving, flying, VR navigation, physical exercise, ...

Setup summary

This device consists of two actuated joysticks on their separately actuated rotational platforms. The joysticks are actuated by Maxon motors (Part number: 268216) and are connected to the handles with steel cables. Booth the joysticks and the platforms are fitted with strain gauges to measure the force exerted by the human. The system is controlled by a Bachmann industrial controller which runs a compiled Simulink model. A desktop pc can communicate with the Bachmann controller trough a TCP or UDP connection.

The setup is used as a bidirectional human machine interface to control simulations (e.g. a tug boat simulator or a Simulation of a walking exoskeleton). The subject holds the two handles with his hands while the system exerts a force or holds a position. This force or position serves as feedback to the subject. The subject can then put the system in a position or exert a force on the handle which serves as the input for the simulation. The subject is not constraint in any way and can let go of the handles at any given point in time.



Risk checklist

Please fill in the following checklist and consider these hazards that are typically present in many research setups. If a hazard is present, please describe how it is dealt with.

Also, mention any other hazards that are present.

Hazard type	Present	Hazard source	Mitigation measures
Mechanical (sharp edges, moving equipment, etc.)	Yes	The handles can exert high forces at the handles. At the stall torques of the motors these forces are F_Z = 192.1 N & F_X = 102 N.	The maximum current to the motors is set in the software with this maximum current the maximum forces are F_Z = 18.2 N & F_X = 9.68 N. The subjects are not physically constraint to the setup. They can let go of the handles on any given time. An emergency stop is present which puts the software in a safe state and disables the motor power through a relay.
Electrical		The device is connected to the main power grid (250AC). In case of component/insulation failure, the user could come in contact with dangerous voltages	All conducting parts of the setup are electrically connected to the earth. The interface touched by the subject is electrically insulated with respect to the rest of the machine.
Structural failure			
Touch Temperature			
Electromagnetic radiation			
Ionizing radiation			
(Near-)optical radiation (lasers, IR-, UV-, bright visible light sources)			
Noise exposure			
Materials (flammability, offgassing, etc.)			
Chemical processes			
Fall risk			
Other:			
Other:			
Other:			

Date 26-10-2017
Contact person Ir. J.B.J. Groot Kormelink, secretary HREC
Telephone
E-mail

Human Research Ethics Committee
TU Delft
(http://hrec.tudelft.nl/)

Visiting address

Postal address

Ethics Approval Application: Human Machine Interface for Exoskeleton
Applicant: Boessenkool, Henri

Dear Henri Boessenkool,

It is a pleasure to inform you that your application mentioned above has been approved.

Good luck with your research!

Sincerely,

Prof. Dr. Sabine Roeser
Chair Human Research Ethics Committee TU Delft

Prof.dr. Sabine Roeser
TU Delft
Head of the Ethics and Philosophy of Technology Section
Department of Values, Technology, and Innovation
Faculty of Technology, Policy and Management

Bibliography

[1] F Hoeckx, A Vrijdag, and D A Abbink. Development of a test setup for exploring the potential of haptic feedback for maritime operations. 2017.

[2] IHMC Robotics. https://ihmcrobotics.github.io/, 2017. URL `https://ihmcrobotics.github.io/`.

[3] Peter D. Neuhaus, Jerryll H. Noorden, Travis J. Craig, Tecalote Torres, Justin Kirschbaum, and Jerry E. Pratt. Design and evaluation of Mina: A robotic orthosis for paraplegics. *IEEE International Conference on Rehabilitation Robotics*, 2011. ISSN 19457898. doi: 10.1109/ICORR.2011.5975468.

[4] Hugo a Quintero, Ryan J Farris, and Michael Goldfarb Members. Control and Implementation of a Powered Lower Limb Orthosis to Aid Walking in Paraplegic Individuals. 19(6):652–659, 2012. ISSN 15344320. doi: 10.1109/ICORR.2011.5975481.Control.

[5] M Vlutters, E H F Van Asseldonk, and H Van Der Kooij. Center of mass velocity-based predictions in balance recovery following pelvis perturbations during human walking. pages 1514–1523, 2016. doi: 10.1242/jeb.129338.

[6] Shiqian Wang, Letian Wang, Cory Meijneke, Edwin Van Asseldonk, Thomas Hoellinger, Guy Cheron, Yuri Ivanenko, Valentina La Scaleia, Francesca Sylos-labini, Marco Molinari, Federica Tamburella, Freygardur Thorsteinsson, Michel Ilzkovitz, Jeremi Gancet, Yashodan Yashodhan Nevatia, Ralf Hauffe, Herman Van Der Kooij, Edwin van Asseldonk, Thomas Hoellinger, Guy Cheron, Yuri Ivanenko, Valentina La Scaleia, Francesca Sylos-labini, Marco Molinari, Federica Tamburella, Iolanda Pisotta, Freygardur Thorsteinsson, Michel Ilzkovitz, Jeremi Gancet, Yashodan Yashodhan Nevatia, Ralf Hauffe, Frank Zanow, Herman van der Kooij, Edwin Van Asseldonk, Thomas Hoellinger, Guy Cheron, Yuri Ivanenko, Valentina La Scaleia, Francesca Sylos-labini, Marco Molinari, Federica Tamburella, Freygardur Thorsteinsson, Michel Ilzkovitz, Jeremi Gancet, Yashodan Yashodhan Nevatia, Ralf Hauffe, and Herman Van Der Kooij. Design and Control of the Mindwalker Exoskeleton. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 4320(c):277–86, 2014. ISSN 1534-4320. doi: 10.1109/TNSRE.2014.2365697. URL `http://www.ncbi.nlm.nih.gov/pubmed/25373109`.