University of Padua

Department of Information Engineering
Master degree in Computer Engineering

Low Obstacles Avoidance for Lower Limb Exoskeletons

Supervisor
Prof. Emanuele Menegatti

Candidate
Edoardo Trombin

Co-Supervisor
Dr. Stefano Tortora

July 18, 2022

Academic Year 2021-2022

Abstract

Powered lower limb exoskeletons (LLEs) are innovative wearable robots that allow independent walking in people with severe gait impairments, or even to augment lower limb capabilities of able-bodied users. Despite the recent advancements, the use of this promising technology is still restricted to controlled research/clinical settings; uptake in real-life conditions as a device to promote user independence is still lacking. The main reason behind this limitation can be traced back to the lack adaptability of LLEs to the different walking conditions that may be encountered in real world settings: the majority of LLEs relies on predefined gait trajectories and is generally unaware of the environment in which gait occurs. This means that the control burden is entirely on the user, resulting in an increased physical and cognitive workload.

This thesis aims at overcoming the aforementioned limitations by proposing a novel approach to enhance the autonomy of the LLEs. In particular, the proposed method has the purpose of estimating the optimal gait trajectory of the exoskeleton in order to autonomously avoid low obstacles on the ground. By using a depth camera coupled with a Computer Vision software module, the environment is sensed to detect the ground plane and obstacles that might interfere with the forward motion, in order to predict the following foothold. Then, an iterative-based collision-free foot trajectory generator

(CFFTG) algorithm is proposed to calculate the optimal foot motion and the joints' angles to be sent to the exoskeleton low-level controllers.
Experimental tests have been carried out in simulation to evaluate both the CV module and the CFFTG based on real data, showing successful performance in different scenarios. In addition, the assumptions that have been considered in this work make the proposed approach compatible with the majority of exoskeletons in research and on the market.
I believe that re-thinking exoskeletons as semi-autonomous agents will represent not only the cornerstone to promote a more symbiotic human-exoskeleton interaction but may also pave the way for the use of this technology in the everyday life.

Contents

List of Figures

List of Tables

List of Acronyms

CFFTG Collision-Free Foot Trajectory Generator.

CoM Center of Mass.

CoP Center of Pressure.

CV Computer Vision.

DC Direct Current.

DoF Degrees of Freedom.

EEG Electroencephalography.

EMG Electromyography.

FSM Finite State Machine.

HT Homogeneous Transformation.

IMU Inertial Measurement Unit.

LLE Lower Limb Exoskeleton.

PCL Point Cloud Library.

PID Proportional Integrative Derivative.

RANSAC Random Sample Consensus.

RGO Reciprocating Gait Orthosis.

ROS Robot Operating System.

1 | Background

1.1 What is an Exoskeleton

A powered exoskeleton is a robotic device that is worn over all, or part, of the human body, powered by a system of electric motors, pneumatics, levers, hydraulics or a combination of cybernetic technologies, to provide ergonomic structural support, while allowing for sufficient limb movement with increased strength and endurance [6]. The exoskeleton is designed to provide better mechanical load tolerance, and its control system aims to sense and synchronize with the user's intended motion. In some applications, exoskeletons are also used to protect the user's shoulder, waist, back and thigh against overload, and to stabilize movements when lifting and holding heavy items [7]. An example of full-body exoskeleton developed to move the limbs of users affected by tetraplegia is showed in figure 1.1.

A powered exoskeleton differs from a passive exoskeleton, as the latter has no intrinsic actuator and relies completely on the user's own muscles for movements, adding more stress and making the user more prone to fatigue, although it does provide mechanical benefits and protection to the user, since it is usually designed using materials, springs or dampers with the ability to store energy from human movements and release it when required. This also explains the difference of an exoskeleton to orthotics, as orthosis mainly aims

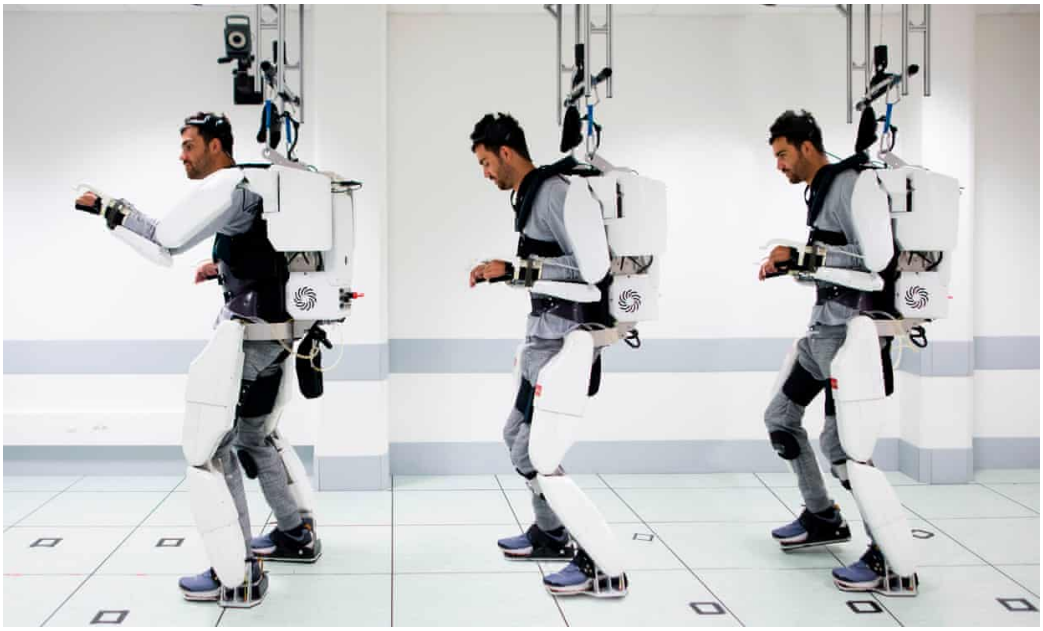


Figure 1.1 Example of full-body Exoskeleton [1]

to promote the progressively increased muscle work and, in the best case, regain and improve existing muscle functions [8, 9].

The general categorization suggests several feasible exoskeleton categories. Such categories have general classes (Figure 1.2), due to the wide quantity of exoskeletons in existence, and are the structure, the body part focused on, the action, the power technology, the purpose, and the application area varying from one to another [10].

1.2 History of Exoskeletons

The earliest-known exoskeleton-like device was an apparatus for assisting movement developed in 1890 by Russian engineer Nicholas Yagin. It used

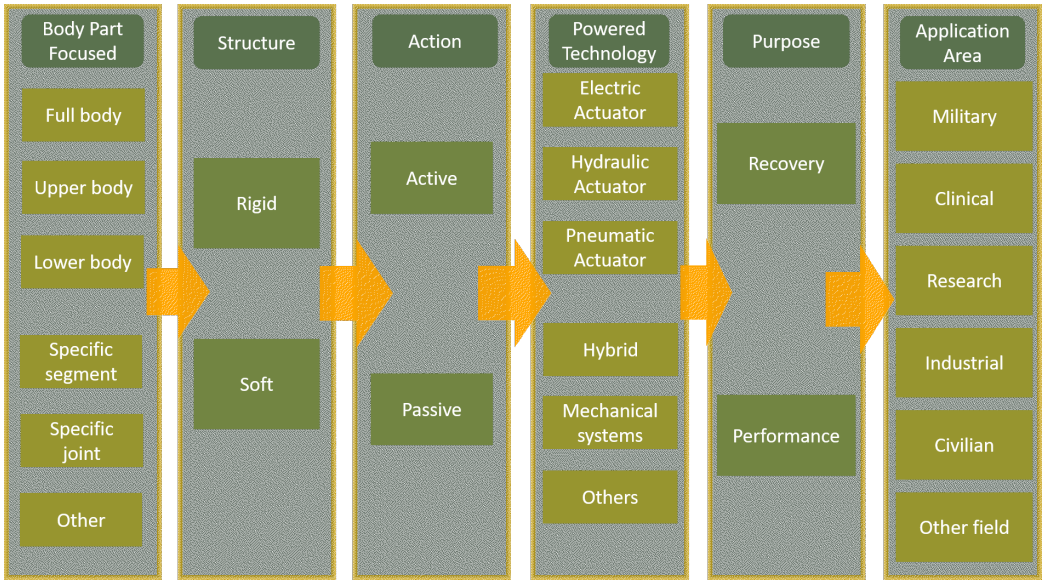


Figure 1.2 General classification model for exoskeletons [2]

energy stored in compressed gas bags to assist in movement, although it was passive and required human power [11]. In 1917, United States inventor Leslie C. Kelley developed what he called a pedomotor, which operated on steam power with artificial ligaments acting in parallel to the wearer's movements. This system was able to supplement human strength with external power [12]. In the 1960s, the first true 'mobile machines' integrated with human movements began to appear. A suit called Hardiman [3] was co-developed by General Electric and the US Armed Forces (Figure 1.3). The suit was powered by hydraulics and electricity and amplified the wearer's strength by a factor of 25, so that lifting 110 kilograms (240 lb) would feel like lifting 4.5 kilograms (10 lb). A feature called force feedback enabled the wearer to feel the forces and objects being manipulated.

The Hardiman had major limitations, including its 680-kilogram (1,500 lb) weight. It was also designed as a master-slave system: the operator was

Figure 1.3 Hardiman Suit [3]

in a master suit surrounded by the exterior slave suit, which performed work in response to the operator's movements. The response time for the slave suit was slow compared to a suit constructed of a single layer, and bugs caused "violent and uncontrollable motion by the machine" when moving both legs simultaneously. Hardiman's slow walking speed of 0.76 metres per second (2.5 ft/s or just under 2 mph) further limited practical uses, and the project was not successful.

At about the same time, early active exoskeletons and humanoid robots

were developed at the Mihajlo Pupin Institute in Yugoslavia by a team led by Prof. Miomir Vukobratović. Legged locomotion systems were developed first, with the goal of assisting in the rehabilitation of paraplegics. In the course of developing active exoskeletons, the Institute also developed theory to aid in the analysis and control of the human gait. Some of this work informed the development of modern high-performance humanoid robots. In 1972, an active exoskeleton for rehabilitation of paraplegics that was pneumatically powered and electronically programmed was tested at Belgrade Orthopedic Clinic [13]. On the same page, ABLE [14], a device that keeps the user standing upright while moving with wheels attached under the feet, was introduced to address the issue of spine compression for wheelchair users. Nonetheless, such simplification in human motion limited navigability. To overcome this, the adaptability and robustness of bipedal walking had to be retained. This required the development of assistive exoskeletons, wearable biped robotic suits that enable paralyzed patients to move with human-like gait patterns. Several lower limb exoskeletons have been developed to resolve this issue and manage to have users walk independently [15, 16, 17, 18, 19, 20, 21, 22]. However, these exoskeletons still lack the full mobility of healthy people, and the motion between the human and the exoskeleton is not yet perfectly synchronized. The development of assistive exoskeletons can be traced back to reciprocating gait orthosis (RGO) [23]. RGO is a passive device that has only one degree of freedom (DoF) in each leg and mechanical constraints that alternatively enable the DoFs. While it has aided users with extra mobility, drawbacks such as long and difficult donning/doffing time, overuse of patients' upper limbs, and required supervision to avoid falling have limited its usage. To address these issues, active walking assisting devices, later known as assistive exoskeletons, were developed. Similar to

RGOs, assistive exoskeletons provide stand-upright ability to users, but also extra DoFs and embedded actuators that allow active control. This results in smoother motion and less energy consumption of the upper limbs.

1.3 Lower Limb Exoskeletons

Lower Limb Exoskeletons (LLEs) are those exoskeletons who are concerned with the movement of lower limbs. There are three major categories of lower limb exoskeletons: augmentation exoskeletons, rehabilitation exoskeletons, and assistive exoskeletons. Figure 1.4 lists many of the lower limb exoskeletons that have been developed in the last 20 years. The horizontal axis shows the development date and the vertical axis the control methods. Each marker shows the active/passive degrees of freedom and actuator types. **Augmentation lower limb exoskeletons** (augmentation exoskeletons) aim at enhancing the physical abilities of healthy users [24]; their design concept can be summarized in one statement: "making the user superhuman". Practically, these exoskeletons aim at reducing the user's energy consumption during walking. For healthy users, predefined trajectories are not necessary. Instead, control algorithms that follow the user's limb motion, such as admittance/impedance control or even positive feedback sensitivity amplification control, are used. Inaccurate but high power/weight ratio actuators, such as series elastic actuators (SEA) and pneumatic actuators, are more commonly used in this category [4]. **Rehabilitation exoskeletons**, on the other hand, are designed to restore abilities such that patients can live without the device. These exoskeletons focus on "how the tasks are done." In most designs, such systems require online adjustments that only "help when necessary" (i.e., assistance-as-needed approach) and reduce assistance as the user gradually improves

[25]. It is expected that the user will regain their lost ability via training with decreasing assistance. The control of this kind of exoskeletons is usually partially predefined since patients need guidance for correct motion profile, but it also adjusts itself based on patient feedback. Portability is usually not considered; the whole system may be fixed to a treadmill under the supervision of a physician [26]. The last category is represented by **Assistive lower limb exoskeletons**, which are those that help users to complete daily activities that they are no longer able to do. For instance, they may assist the user to walk when they are normally unable to due to spinal cord injury, stroke, or age deterioration.

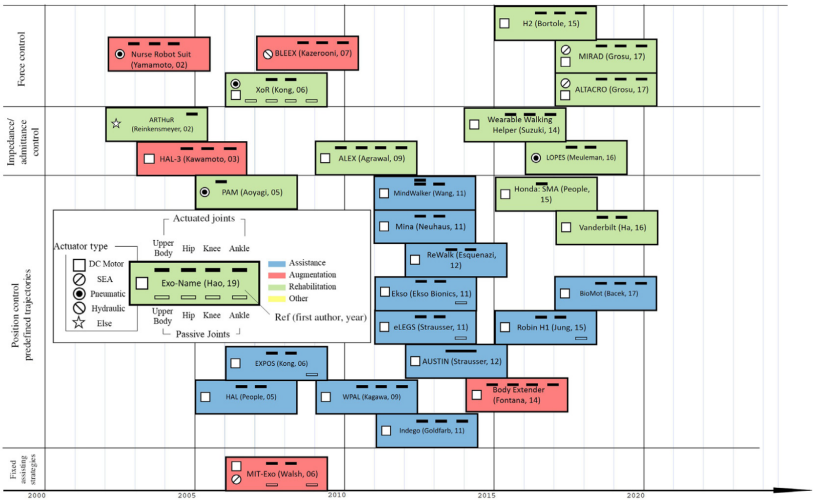


Figure 1.4 A temporal overview of LLEs research [4]

Assistive exoskeletons are mostly used by thoracic-level motor-complete spinal cord injury (SCI) patients. Many of these patients permanently lose the ability to walk and consequently use wheelchairs. However, the accessibility of the wheelchair is limited; common environments such as stairs are not navigable or require extra assistance. Additionally, research has found that remaining seated for long periods induces health issues. It is suggested that

passive mechanical loading is necessary for maintaining bone mineral density (BMD). BMD of long-time wheelchair users is statistically lower than that of individuals who stand with assisting tools [27]. Wheelchair users also suffer from pressure sores and ischial tuberosities since a high amount of pressure is applied on the seating surface for long durations [28]. As a result, assisting devices that keep users standing upright with better maneuverability, are required.

Assistive exoskeletons are often controlled with predefined trajectories triggered by the user's moving intention [29, 30, 31, 15, 16, 17]. High precision control is required; most assistive exoskeletons are driven with DC motors. The control is generally a combination of a high-level central controller and low-level joints controllers. Since most assistive exoskeletons users have lost the ability to move one or more limbs, the motion is entirely generated and carried out by the exoskeleton, while the user sends high-level commands (e.g., sit, stand, move forward, stop, turn) to control the general behaviour of the exoskeleton. To capture high-level commands, exoskeletons need to estimate the user's intention. The easiest method is having the users manually input the command. A common design is placing a console besides the user's wrist. However, this will limit users' upper limbs and cause delay since commanding legs by hands is not intuitive. Direct estimation from biological signals has been developed, yet biological signals are often noisy and difficult to measure. More accessible signals such as EMG are not usable because paralyzed patients have often lost the connection between the brain and the limbs. Thus, indirect measurement has been introduced [19]. Additional sensors are added, such as inertial measurement unit (IMU) or ground reaction force (GRF) sensors. It is known that the center of mass (COM) velocity is highly related to the intention of motion [32]. For example, COM

forward movement is associated with walking forward. With IMU located near the torso or force sensor under users' feet, the motion of CoM can be estimated. In addition, posture can also insinuate the moving intention. By measuring the torso angle, the exoskeleton can tell whether the user wants to initiate forward motion. These sensors do not need high sample frequency or filtering since they only provide a rough estimation of human intention. Some lower limb exoskeletons, such as **ReWalk** [15, 16], use a control panel and tilt sensor. Others, such as **Robin H1** [33], estimate the user's intention by measuring the center of pressure (CoP). To increase comfort and stability, instead of developing the exoskeleton with a rigid human robot interface and position control, **Mina** [17] and **Indego** [34, 35] add a compliant assisting mode to partially preserve users' walking ability. **BioMot** [36] developed a novel variable stiffness actuator to further control the torque and stiffness at each joint. After the task is determined, the exoskeleton will generate the required joint force/position profile to accomplish the task. These trajectories are generally pre-programmed and linked to specific motions. In most cases, trajectories are generated with a finite state machine. A finite state machine is a controller that divides the single full motion cycle into different phases. For each phase, the controller may have a different control scheme. This is due to the natural discontinuity in lower limb motions such as bipedal walking [37] because different phases of walking present different dynamics. For example, the model of single support (with only one foot contacting the ground) can be viewed as an inverted pendulum, while the double support phase (both feet are on the ground) cannot. It is important to know that the generated trajectory may not be identical to a healthy subject's motion profile, but it can still complete the task. For example, one feature that is commonly seen in many assistive exoskeletons is passive or fixed ankle joints.

While humans rely on ankle push-off torque as the main thrust during walking, researchers have shown that actuating only the hip joints can create stable gaits for passive walkers [38, 39]. Thus, for individuals with complete loss of motor skills, human-like gait patterns can be replaced with simpler gait patterns and fewer actuated joints to reduce exoskeleton weight and complexity. Despite the fact that the human lower limbs are 12-DoF mechanisms (for each leg, three at the hip, one at the knee, and two at the ankle), most assistive exoskeletons exclude the DoF in the transverse plane and active DoF at the ankle. Instead of purely relying on predefined joint trajectories, some assistive lower limb exoskeletons generate motion profiles with online calculation. **Wearable Power-Assist Locomotor** (WPAL) [40, 41] generates the trajectories based on "not falling back" and "provide ground clearance". Desired hip and toe positions are given, and trajectories are generated with minimum jerk. **MindWalker** [42, 43] also applied a similar idea, but only during the weight-shifting phase (double support phase), and the predefined trajectories are corrected online if the exoskeleton senses imbalance. While many lower limb exoskeletons control each joint individually, the relationship between different joints has been studied. Researchers have found that the motion of the knee and hip are coupled due to bi-articular muscle connections. **AUSTIN** [31] takes advantage of this coupling and reduces the requirement of actuation by using a single actuator for hip and knee joints. Most assistive exoskeletons require extra support (e.g., crutches, rollator) to maintain user's balance during motion. MindWalker [42, 43] addresses this issue by estimating the XCoM (the position of the center of mass combined with momentum, during walking) to prevent falling, yet full self-balancing is not accomplished. One of the exceptions is REX, a lower limb exoskeleton developed by REX Bionics in New Zealand, which moves slowly to statically

balance itself with only two legs [18]. Nevertheless, for bipedal systems to maintain dynamic balance, the properties of human limbs, such as center of mass, inertia, and link length need to be precisely measured, which is still a challenge with current technology. Table 1.1 summarizes the principal features of the assistive Lower Limb Exoskeletons mentioned in this section.

A major challenge related with this category of Lower Limb Exoskeletons lies in the fact that assistive exoskeletons are supposed to help disabled people in their daily activities, which are often carried out in uncontrolled settings which may span through a plethora of different environments. In addition, as previously mentioned, the user has little direct control on the exoskeleton since its interactions are limited by its clinical condition. Thus, to be effectively used in these conditions, assistive lower limb exoskeletons should be able to adjust the gait pattern according to the different types of environments, either fully or partially autonomously, e.g. by following additional high level commands given by the user. To overcome these limitations, this thesis fits into a recent line of research that aims at enhancing the level of autonomy of assistive Lower Limb Exoskeletons through methods and techniques that are commonly employed in artificial intelligence and autonomous robotics [44].

Assistive Exoskeletons			
Name	**Degrees of Freedom**	**Intention Estimation Method/Trajectory Generation Method**	**Actuator Type**
eLEGS [29]	6, 4 actuated (hips and knees), passive ankle joints	1. Force sensors on foot pads and crutches, IMU on arms for estimate the arm angle. Measurements are fed into finite state machine to estimate the walking stage. 2. Predefined joint trajectories related to the finite state machine to assist bipedal walking and sit-to-stand.	DC motors
Ekso [30]	6, 4 actuated, 2 hips and 2 knees. 2 passive ankles		
AUSTIN [31]	4, 2 actuators, hip and knee joints are coupled	Predefined joint trajectories based on clinical gait analysis	DC motors
ReWalk [15, 16]	6, 4 actuated, 2 hips and 2 knees, passive ankles	Function selector on forearm and tilt sensor on torso to trigger predefined motion profile	DC motors

Assistive Exoskeletons			
Name	**Degrees of Freedom**	**Intention Estimation Method/Trajectory Generation Method**	**Actuator Type**
Mina [17]	4, 2 hips and 2 knees	Predefined joint trajectories based on healthy subject wearing Mina. Can switch between rigid position control mode and compliant assist mode	DC motors
REX [18]	5 actuated each leg	N/A	DC motors
HAL [19, 45, 46]	4 actuated, 2 hip and 2 knees	1. Myoelectric sensors 2. Gyroscope and accelerometer on torso/-ground reaction force (GRF) sensors 3. Assist sit-to-stand motion, bipedal walking 4. Estimate walking speed and generate desired trajectories with inverse kinematics	DC motors
ROBIN H1 [33]	6, 4 actuated, 2 hips and 2 knees, 2 passive ankles	IMU, encoders and foot sensors (can be replace with neural network-based classifier with IMU and encoders' measurements)	DC motors

Assistive Exoskeletons			
Name	**Degrees of Freedom**	**Intention Estimation Method/Trajectory Generation Method**	**Actuator Type**
WPAL [40, 41]	6 actuated with a walker	1. Trigger with angle-acceleration sensor on the walker and foot pressure sensor 2. Calculate minimum jerk trajectories with desired toe position	DC motors
Mind Walker [42, 43, 47, 48]	6 actuated, 4 hips and 2 knees	1. Estimate CoM position with IMU located near hip 2. Swing stage: Modified joint trajectories recorded from healthy subjects and online correction 3. Weight shifting: Interpolation between start and end points	DC motors with series elastic actuators
Indego/ Vanderbilt [34, 35]	4 Actuated, 2 hips and 2 knees. Standard ankle orthosis	Using joint angles to estimate the distance between CoP and forward foot to trigger finite state machine	DC motors

Assistive Exoskeletons			
Name	**Degrees of Freedom**	**Intention Estimation Method/Trajectory Generation Method**	**Actuator Type**
BioMot [36]	6 actuated on sagittal plane (hips, knees, and ankles)	1. During ankle push-off and heel- off, actuators provide fixed ramp torque assistance 2. The rest of the time the interacting force between the exo and the user is minimized	Variable stiffness actuators
EXPOS [49]	4 actuated (hips and knees), 2 passive (ankles), flexible frame for transverse, frontal plane motion	1. Pressure sensors are installed at human-exoskeleton interacting points, the measurement will increase due to muscle contraction 2. Generate assistive torque proportional to the contact pressure difference between the front and rear sides of the leg	Motor-cable driven

Table 1.1 Assistive Lower Limb Exoskeletons

1.4 Related Work

This section will review some lower limbs exoskeletons' projects that have the goal of surpass the limits of assistive Lower Limb Exoskeletons presented in the previous section. To overcome the limitation posed by the predefined behaviours that are found in most commercially available exoskeletons, solutions that employ additional information sent by the users (mostly high-level commands obtained by processing bio-signals) have been exploited. [42, 43, 47, 48] focuses on the correct estimation of users' intention thanks to the combined use of electroencephalogram (EEG) and EMG signals with the goal of producing customised online leg trajectories by using Model Predictive Control (MPC) coupled with the high-level signals previously acquired. In a similar fashion, [50] combines the use of EEG and EMG signals acquired from the user to define a reliable set of high-level commands used to effectively climb stairs. In this solution, EEG signals are used to determine the leg that has to be controlled (left or right), while EMG signals are used to determine the parameters of the stair (height and width of the steps). [51] makes use of surface EMG (sEMG) signals sent by the user to determine the step height during the gait, by classifying the signals through the use of a back propagation neural network. [52] makes a step further in the direction of the correct estimation of user's intention through the use of sEMG signals collected from lower limbs. In this work, different classification methods are evaluated with the goal of finding the one that gives the most accurate predictions on the type of surrounding environment (level ground, ascending and descending ramps and stairs). Also, changes in sEMG signals in presence of muscle fatigue are studied, since these changes are an issue that is often overlooked when developing an sEMG classifier.

Although the solutions exploiting bio-signals seem to work well in most cases, a lot of complications come with the correct estimation of users' intention, as [52] has demonstrated in the specific case of muscle fatigue. Besides the fact that, to control an exoskeleton in such a way, the user must be trained for a fair amount of time, it also causes additional burden to the user, which will have to be aware of the signals that he/she is consciously and unconsciously sending to the exoskeleton.

A different approach to the problem of predefined behaviours lies in the addition of exteroceptive sensors to the exoskeleton, that can collect information about the environment, coupled with algorithms that produce specific exoskeleton's behaviours based on such information. [53] considers the problem of stair ascending for Lower Limb Exoskeletons by using a depth camera and RANSAC to identify the stairs, and then calculate inverse kinematics to compute the actual motion after the stairs' geometry is computed. [54] employs two time-of-flight laser range sensors placed on the feet and dynamic movement primitives (DMP) theory to climb stairs. First, stairs are modeled according to the range sensors' measurements, and then DMP, which is a bio-inspired method that models complex motion tasks in a simple and elegant way, produces a trajectory by considering the horizontal surface of the steps as goals, and step edges as obstacles. [55] considers the problem of terrain classification by using a camera attached to the waist to collect images that are then fed to a Convolutional Neural Network (CNN) that outputs the prediction (level ground, inclined and declined stairs). [56] expands the capabilities of the previously described work, since it also extracts additional terrain-specific parameters (slope in the case of ramps, step height and width in the case of stairs) that will then be used to produce a specific gait appropriate for the predicted terrain. While the results are quite promising, the

major limitation of these approaches is the lack of generalization that is introduced when focusing on specific models such as stairs or ramps, since the models that have been used in the aforementioned cases are not capable of dealing with all types of obstacles that can be stepped on, and have height and width similar to stair steps; small obstacles that can be safely stepped over are also not considered in these works. Nonetheless, a movement towards generalized models can be noticed, since [56] already refers to the exploited terrain models as "specific types of obstacles", which is an appropriate term which implies an underlying general categorization that is the ultimate aim of exoskeleton's obstacle detection and terrain classification. VALOR [57] is one of the latest researches which couples assistive LLEs and computer vision, and aims at solving the task of low obstacle avoidance. The gait planning problem in presence of low obstacles is divided in the following sub tasks of obstacle detection and gait pattern planning. The first sub task is solved through the use of a Computer Vision algorithm and with the assistance of a depth camera, to find geometrical parameters of all obstacles, such as length, width, height and distance from the exoskeleton (calculated with respect to the nearest foot). The second sub task is solved by selecting the correct gait phase, step length and height, and applying inverse kinematics to find the joint angles that produce the desired trajectory, given obstacles information obtained by solving the previous sub task. Even if this solution has many advantages, like treating low obstacle avoidance in a simple and elegant way and without adding unnecessary complexity to the task, it also poses some limitations. For example, the trajectories computed for the foot that will step over obstacles are fixed to have the maximum height in the middle between initial and final position. This seems unnatural, since humans are able to adapt the step trajectory in a much more unconstrained way. Additionally,

the obstacle detection routine extensively evaluates all obstacles, without posing the problem of the distance between them and the exoskeleton, which is considered later, when extensive geometric evaluations have already been done for all obstacles. In a real walking situation, a human is only concerned with those obstacles that will affect his motion; if an obstacle is too far away, its importance becomes negligible.

1.5 Thesis Aims and Structure

To summarize the content of this chapter, exoskeletons are a break-through technology for physically impaired persons, but still lack the synergy with the user and the environment that would be required to allow a safe use in unconstrained home environments. In the midst of this challenging scenario, this thesis aims to overcome some of the aforementioned limitations by proposing a novel vision-based autonomous control of lower limb exoskeletons with the specific purpose of traversing small obstacles. The work will be carried with the idea of enhancing the autonomy of the exoskeleton, rather than the perception of users' signals or the estimation of users' intention, in order not to increase the burden on the user. In particular, in this thesis I proposed and implemented:

- a light-weight computer vision-based algorithm to recognize low obstacles crossing the walking path and to determine the most appropriate foot position according to the current scenario;

- a novel Collision-Free Foot Trajectory Generator (CFFTG) algorithm to compute the optimal gait trajectory to place the foot in the desired position while avoiding collisions with the obstacle.

a The proposed method has been tested in simulation with data acquired in real-world scenario and a direct comparison has been done with respect to VALOR's proposed solution, since they both focuses on the same problem and application.

The dissertation will continue as follow: Section 2 provides a detailed presentation of the problem and the proposed solution, as well as the description of the experimental setup for evaluation. The results of the experiments are shown in Section 3 and discussed in Section 4. Finally, Section 5 provides some conclusive remarks and future developments.

2 | Methods

2.1 Problem Description

As previously discussed, assistive Lower Limb Exoskeletons have the potential to be a step ahead of other assistive devices, such as wheelchairs, since they can expand the capabilities of disabled users to move more freely and naturally throughout the environment. Nevertheless, almost all of the current assistive Lower Limb Exoskeletons being developed still have huge limitations, one of them lying in the predefined nature of the trajectories chosen by the exoskeletons' low-level controller. Although this may just seem a reasonable way of solving the problem of motion, it poses a series of complications, which include the users' discomfort, given by the fact that these trajectories are usually not physiological, and the inability to move efficiently in complex environments. One of the goals of this thesis is to overcome such limitation by providing customised trajectory that better fit the environmental condition in which the exoskeleton is moving, while considering the most general type of assistive Lower Limb Exoskeletons, meaning that all the (numerous) limitations mentioned in the previous chapter will be considered. More specifically, the following **assumptions** will be made:

1. An assistive exoskeleton similar to the ones previously described in section 1.3 will be used. It will come with crutches to maintain balance

and will only have PID controllers to interact with joints' motors (other hardware assumptions are discussed in section 3.1);

2. The exoskeleton will move inside an **indoor environment**;

3. The exoskeleton will be only concerned with the task of moving forward (through bipedal locomotion). This assumption is made since turning requires complex manoeuvres by the user, in contrast with forward motion, that can be executed by the exoskeleton almost autonomously (the only interaction needed in this case is a stop / start command that can be sent by the user);

4. Availability of a **depth camera** that can be attached to the waist of the user and will be used to perceive the surrounding environment.

In this specific context, the present work is concerned with the avoidance of low obstacles that can be encountered during forward motion. Since is "low obstacle" is not a quantitative definition, a reasonable assumption for low obstacles is that their maximum height shouldn't exceed the maximum height of a stair step. Italian decree n. 236 enacted the 14th of June, 1989 [58] defines a stair step height range between 16 and 20 cm. It will be therefore assumed that the low obstacles mentioned in the present work will have an height not exceeding **20 cm**.

2.2 Overview

The pipeline developed to execute the task of low obstacle avoidance is the following:

- Collect environmental information in the form of point clouds through the use of the depth camera;

- Detect obstacles, and based on the detection, select the best next step length;
- Based on the output of the previous phase, a trajectory generation algorithm will find a collision-free trajectory for the foot in the sagittal plane;
- Given the foot trajectory, an inverse kinematics solver will find joint angles and velocities that meet the kinematics costraints. These angles and velocities will then be fed to the PID controllers to produce the actual movement of the exoskeleton.

A scheme of the proposed solution is displayed in figure 2.1.

2.3 Computer Vision Module

In this subsection the proposed computer vision module will be discussed. As mentioned in section 2.1, this module expects a point cloud as input, and outputs parameters regarding the best length for the next step and the obstacles between the current exoskeleton's position and the chosen foothold position. This will be done for each foot. The pipeline of the module is detailed in Algorithm 1.

Algorithm 1 Computer Vision Module

Input: cloud, userP
Output: obstaclesShape, stepLength
filteredCloud ← **downSampling**(cloud)
modelCoeffs, plane, obstacles←**groundPlaneDetection**(filteredCloud)
homogeneousTransformation(modelCoeffs)
tracks, obstaclesShape ← **tracksEvaluation**(plane, obstacles, userP)
stepLength ← **footHoldEvaluation**(tracks, userP)
return obstaclesShape, stepLength

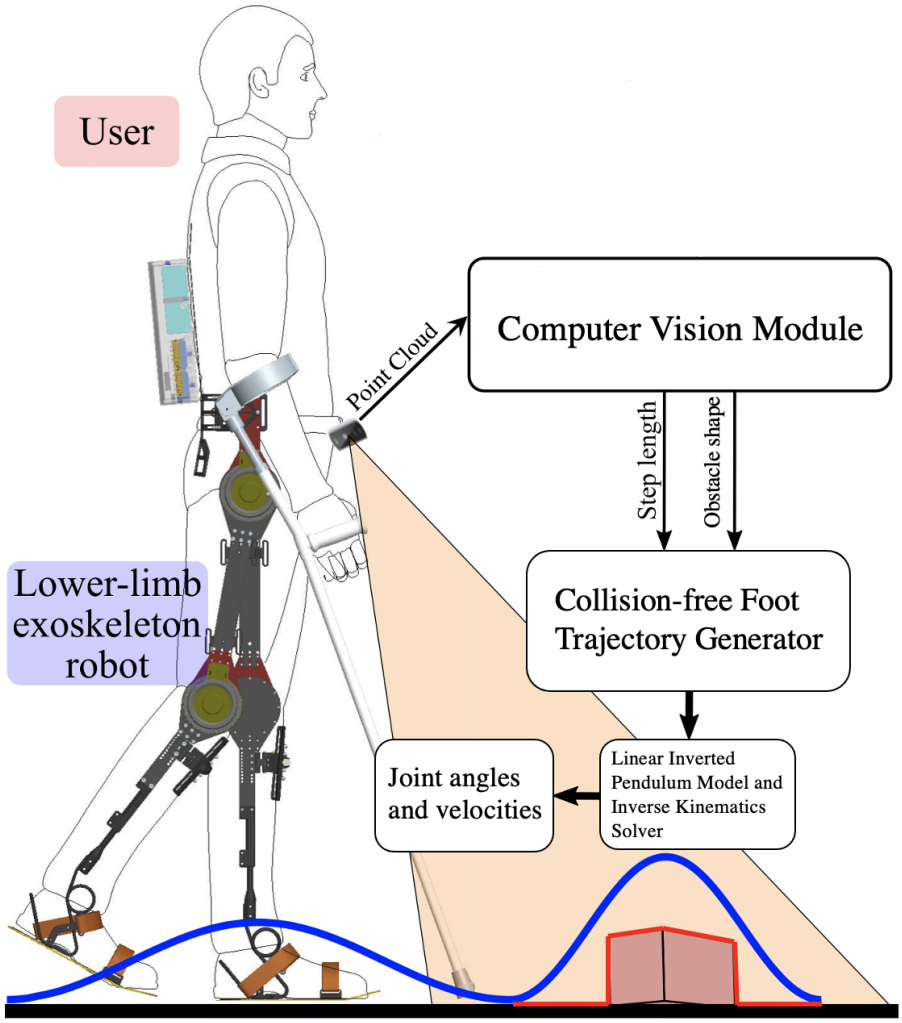


Figure 2.1 A general scheme of the proposed solution.

The following sections will follow module's pipeline, from input to output.

2.3.1 Input Point Cloud

A point cloud, by definition, is a set of data points in space. This definition implies that each point is expressed in 3D coordinates, and eventually some

additional information. Figure 2.2 shows an example of input point cloud visualized in RViz (one of ROS main visual tools, along with Gazebo).

Figure 2.2 An example of input Point Cloud visualized in RViz

2.3.2 User Parameters

User parameters are necessary to perform calculation involving foot placement. This bundle of data must include:

- Maximum step length (in cm), defined as the maximum distance between the COM and the next step foothold that is being calculated;
- Distance between feet in the frontal plane, needed to perform Track Evaluation (see section 8;
- Foot length and width;
- Duration of a step;

- A parameter *safeDist* that represents the minimum distance between foot and obstacle. In the present work the parameter has been experimentally determined by tests on the Collision-Free Foot Trajectory Generator Algorithm (more details on this function can be found in section 2.3.7).

In pseudocode, user parameters will be referred to as `userP`, and will be assumed that `userP` is a structure containing the aforementioned values.

2.3.3 Filtering

The point cloud comes as a very dense set of points, with most neighbouring points having distance less than a centimeter (even considering that as the distance from the camera increases, the distance between neighboring points also increases). This condition happens to be excessive in the case of low obstacles avoidance, since very small obstacles (with a volume $\neq 1$ cm^3) do not have enough inertia to affect exoskeleton's forward motion in a significant way. It is also computationally more demanding to process such a huge set of points. That's why, before performing the actual computations, the point cloud is down-sampled to output a smaller set, trying to lose as little information as possible. To do so, a Voxel grid is created. In the same way a digital photo is represented by a 2D Pixel grid, a Voxel grid is a 3D grid that can discretize a point cloud. In practice, a set of points inside a Voxel (element of the 3D grid, namely a cube, also called leaf) is approximated by a single point by taking their centroid. In such a way, the size of the cloud is reduced without losing much information about the original cloud. A Voxel edge length of 2.5 cm has been experimentally chosen by observing the changes of the cloud after the down-sampling . This value happened to provide a significant reduction of the set, while maintaining a rich representation of

the environment.

Algorithm 2 DownSampling

```
Input: cloud
Output: filteredCloud
voxelGrid ← createVoxelGrid( cloud )
setLeafSize( voxelGrid, 2.5)
filteredCloud ← filter( voxelGrid )
return filteredCloud
```

2.3.4 Ground Plane Detection

After down-sampling the point cloud, the next step consists of detecting the ground plane points, so that we can distinguish them from the obstacle points. The algorithm used to accomplish this task is Random Sample Consensus (RANSAC) [59]. RANSAC is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers do not influence the values that it estimates. Therefore, it can also be interpreted as an outlier detection method [60].
A generic RANSAC algorithm is shown in Algorithm 3.

The functions **CalculateParamsFromSet** and **CalculateError** are selected according to the specific problem, since they depend on the specific model and loss function used.
In the case of an indoor ground plane, the set of points belonging to it would form an ideal 3D plane, therefore the algorithm will look for a model with equation $ax + by + cz + d = 0$, trying to find the values of a, b, c, d that best fit the data set. In practice, the best values will be those that have a sufficient number of points fitting the plane (inliers) with a small error. The specific RANSAC implementation used for this thesis is the one found in Point Cloud Library (PCL) [61]. The Ground Plane Detection function is

Algorithm 3 RANSAC

Input: dataset, model, maxIterations, threshold, initSetSize, minSetSize
Output: bestModelParams, bestInliers

```
iterations ← 0
bestModelParams ← NULL
bestModelError ← +∞
bestInliers ← { }
while iterations < maxIterations do
  initialInliers ← {initSetSize points extracted at random from dataset}
  modelParams ← CalculateParamsFromSet(initialInliers)
  additionalInliers ← { }
  for every point in dataset not in initialInliers do
    if CalculateError(point, modelParams) < threshold then
      add point to additionalInliers
    end if
  end for
  if size(initialInliers) > minSetSize then
    inliers ←merge(initialInliers, additionalInliers)
    updatedModelParams ← CalculateParamsFromSet(inliers)
    modelError = CalculateError(inliers, updatedModelParams)
    if modelError < bestModelError then
      bestModelError ← modelError
      bestModelParams ← updatedModelParams
      bestInliers ← inliers
    end if
  end if
  iterations ← iterations + 1
end while
return bestModelParams, bestInliers
```

detailed in Algorithm 4.

Algorithm 4 groundPlaneDetection

Input: filteredCloud
Output: modelCoeffs, plane, obstacles

dataset ← filteredCloud
model ← $ax + by + cz + d$
maxIterations ← 3000
threshold ← 2 cm
initialSetSize ← 3
minSetSize ← **size**(filteredCloud)/2
modelCoeffs, plane ← **RANSAC**(dataset, model, maxIterations, threshold, initSetSize, minSetSize)
obstacles ← filteredCloud \ inliers #set difference
return modelCoeffs, plane, obstacles

2.3.5 Homogeneous Transformation

An aspect that wasn't addressed in the previous sections is that not every point cloud can be used. When the exoskeleton performs bipedal walking, it switches between double support phase (when both feet are in contact with the ground) and single support phase (when a single foot is in contact with the ground). In single support phase, the exoskeleton is executing a step, and has no need to plan the next until it reaches the next double support phase. It is therefore best to only use point clouds collected during double support phases. The tilt angle (displayed in figure 2.3) measured at initial conditions will be roughly the same measured in any double support phase, so comparing the initial tilt angle (that we can consider given since it relies one the same routine shown in this section) with the successive ones will give us information about the support phase for every cloud. Apart from that, there is a plethora of ways to acknowledge the fact that the exoskeleton is in double support (contact sensors placed on the feet, IMU to measure the

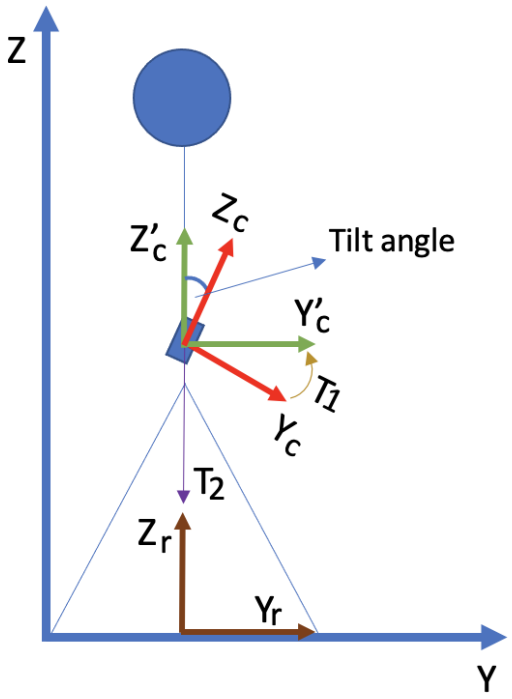


Figure 2.3 Transformations between frames

velocity of the CoM, inverse kinematics, etc...) Therefore, in this work we will always assume that the algorithm works when the exoskeleton is in double support. After obtaining the tilt angle it is also useful to switch from camera coordinate system (also called frame) in which cloud points are originally represented, to robot coordinate system, which lies under the camera, where exoskeleton's feet and ground come in contact. Figure 2.3 shows all of the principal elements involved:

- The red arrows represents the Z-axis and Y-axis of the camera coordinate frame;

- The green arrows represents the Z-axis and Y-axis of the camera coordinate frame when rotated to be aligned with the robot frame;
- The brown arrows represents the Z-axis and Y-axis of the robot coordinate frame;
- X-axis for all frames is orthogonal to the other two axis of that frame and, although not shown in the picture, can be found by using the right-hand rule.
- Camera frame is rotated about the X-axis with respect to the robot frame. This is what was previously called tilt angle.

Given $\vec{n_{z_c}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$ the Z-axis of the robot frame, and $\vec{n_{z_r}} = \begin{bmatrix} a & b & c \end{bmatrix}$ the normal vector to the ground plane previously detected in camera coordinates. The value of $\vec{n_{z_r}}$ has already been computed in the previous section, and is stored inside the variable `modelCoeffs`. The dot product of $\vec{n_{z_c}}$ and $\vec{n_{z_n}}$ can then be computed as:

$$dot = \vec{n_{z_c}} \cdot \vec{n_{z_r}} = (0 \times a) + (0 \times b) + (1 \times c) = c \tag{2.1}$$

Then the geometric definition of dot product can be applied to obtain the tilt angle:

$$\vec{n_{z_c}} \cdot \vec{n_{z_r}} = \|\vec{n_{z_r}}\| \|\vec{n_{z_c}}\| \cos(tiltAngle) \tag{2.2}$$

$$tiltAngle = \arccos\left(\frac{dot}{\|\vec{n_{z_r}}\| \|\vec{n_{z_c}}\|}\right) = \arccos\left(\frac{c}{\sqrt{a^2 + b^2 + c^2}}\right) \tag{2.3}$$

The cloud points coordinate system can be thus changed by using the concept of Homogeneous Transformation (HT), a projective geometry concept that allows to map points between different coordinate frames, when the mathematical relationship between them can be represented as a combina-

tion of rotation and translation [62]. As already stated, the two frames differ by a rotation about the X-axis; they also differ by a translation whose value is the distance between camera and ground. Performing the homogeneous transformation implies finding the 4 x 4 homogeneous transformation matrix ${}^{C}_{R}T$ that represents the mapping between camera and robot frames. ${}^{C}_{R}T$ will have the following form:

$$ {}^{C}_{R}T = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2.4} $$

where R is a 3 x 3 matrix(called rotation matrix), and t is a 3 x 1 vector(called translation vector). The transformation between cloud points in camera coordinates $\vec{p_c} = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}$ and robot coordinates $\vec{p_r} = \begin{bmatrix} x_r & y_r & z_r \end{bmatrix}$ can then be expressed as:

$$ \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix} = {}^{C}_{R}T \cdot \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{2.5} $$

To simplify the problem, a property of homogeneous transformation has been exploited: subsequent homogeneous transformations can be chained by multiplying the two transformation matrices. The opposite is also true, meaning that we can decompose a transformation into a chain of transformations that produce the original one when multiplied. The problem can then be decomposed into two steps:

- Find the rotation matrix that aligns the vector normal to the ground

plane to the Z-axis of the robot coordinate frame, then perform a rotation-only transformation (expressed by T_1);

- After performing the rotation-only transformation, calculate the translation vector between camera and robot frame coordinates and perform a translation-only transformation (expressed by T_2) to complete the alignment.

${}^{C}_{R}T$ will be then decomposed as:

$$ {}^{C}_{R}T = T_1 \cdot T_2 = \begin{bmatrix} R_T & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} I & t_T \\ 0 & 1 \end{bmatrix} \tag{2.6} $$

To find the rotation matrix R_T that rotates $\vec{n_{z_c}}$ into $\vec{n_{z_r}}$, it is necessary to:

- Calculate *dot* (already done) and

$$ \vec{cross} = \vec{n_{z_c}} \times \vec{n_{z_r}} = \begin{bmatrix} b & -a & 0 \end{bmatrix} $$

by applying the definition of cross product;

- Define the skew-symmetric cross-product matrix of $\vec{cross}$:

$$ S = \begin{bmatrix} 0 & -cross_3 & cross_2 \\ cross_3 & 0 & -cross_1 \\ -cross_2 & cross_1 & 0 \end{bmatrix} \tag{2.7} $$

- Obtain rotation matrix R_T as:

$$ R_T = I + S + \frac{S^2}{1 + dot} \tag{2.8} $$

At this point the rotation-only transformation can be performed, rotating $\vec{n_{z_c}}$ onto $\vec{n_{z_r}}$. The last step is the translation-only transformation needed to set the origin of the frame on the ground. In this case it is enough to take a random point stored inside the `plane` array (which will already be rotated) and measure its Z coordinate, t_Z. Its absolute value will be the height of the camera with respect to the ground.

The translation vector t_T will then be $t_T = \begin{bmatrix} 0 \\ 0 \\ -t_Z \end{bmatrix}$.

The pseudo-code implementation is shown in Algorithm 5.

Algorithm 5 homogeneousTransformation

Input: modelCoeffs
Output: tiltAngle

dot ← modelCoeffs[2]
tiltAngle ← **arccos**(dot / **norm**(modelCoeffs))
cross ← {modelCoeffs[1], -modelCoeffs[0], 0}
S ← **createSkewMatrix**(cross)
R_T ← **assembleRotationMatrix**(skewMatrix, dot)
t_T ← {0, 0, 0}
T_1 ← **createHTMatrix**(R_T , t_T)
applyHT(T_1, {plane, obstacles})
randPoint ← { random point extracted from plane}
t_Z ← randPoint[2]
t_T[2]← -t_Z
T_2 ← **createHTMatrix**(I , t_T)
applyHT(T_2, {plane, obstacles})
return tiltAngle

2.3.6 FootHold Identification

After processing the point cloud from the camera, the pipeline will give a score to all possible foothold points on the ground plane. Since we are in-

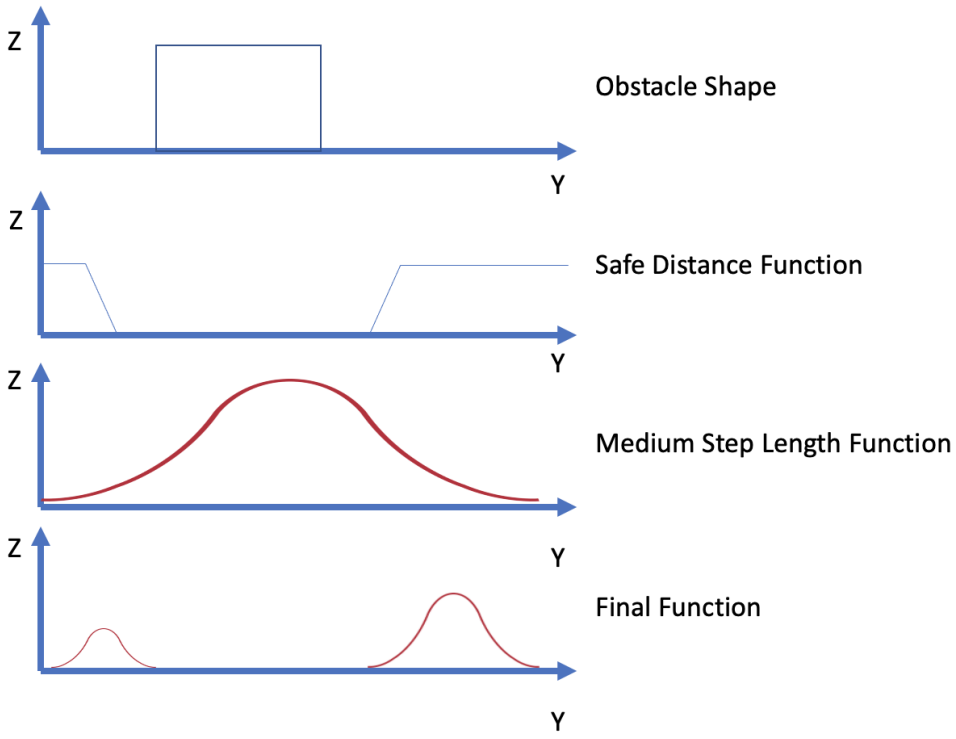


Figure 2.4 Functions involved in Tracks Evaluation

terested in a forward motion without turns, the possible foothold points will have a double-track shape, with one track for each foot. This evaluation will be mainly affected by the distance with respect to the nearest obstacle point and the step length (privileging steps of medium length) (see image 2.4).
As mentioned in section 2.3.2, a function is used to relate the obstacle height and the minimum distance required to step over it safely. This implies that we have to evaluate obstacle points nearby each track first, and then decide the appropriate minimum distance. In this way, we can also produce a sagittal plane (Y-Z plane) image of the obstacle points inside each track (variable `obstacles_shape`). This object will then be fed into the Collision-Free Foot

Trajectory Generator algorithm to find the optimal foot motion. This routine is computed for each of the two tracks, each one representing one foot. Even if in the actual gait, only one of the two tracks is evaluated at each iteration of the Computer Vision algorithm (successive steps clearly alternate between feet), we assume to always be in the ”worst” case, which is the first step. When planning the first step, we can start with either foot, so the evaluation has to be computed for both tracks. Nonetheless, the algorithm can be easily modified to alternate between tracks after the first step.
The implementation of the function is shown below (Algorithm 6). First off, each obstacle point is evaluated. Function **insideTracks** verifies that a point is in possible collision with one of the two feet by measuring the X and Y coordinates of the point.

Algorithm 6 insideTracks

```
Input: point, userP
Output: type
if point.y < userP.maxStepLength and
userP.distBtFeet/2 < || point.x || < userP.distBtFeet/2 + userP.footWidth
then
  if point.x < 0 then
    type ← 0   #left track
  else
    type ← 1   #right track
  end if
else
  type ← -1   #outside tracks
end if
return type
```

Then, the obstacle Y-Z discrete function is created for each track, by storing Y-Z pairs. If two points have the same Y, the one with the highest Z coordinate is stored. The maximum obstacle height for each track is also evaluated.

Afterwards, the minimum distance for each track is set (**setMinDist**) by evaluating the function stored inside `userP` (section 2.3.2) which relates minimum distance and maximum obstacle height. The last and most crucial part of this routine is the score evaluation for each plane point inside the tracks. This calculation only takes place if the maximum obstacle height for each track is lower than 20 cm. As mentioned in section 2.1, an obstacle is considered low when its height is lower than 20 cm, so detecting an obstacle taller than that automatically makes the next steps unfeasible. It doesn't matter if the subsequent step is actually feasible, since the forward motion will eventually require to step over the obstacle after few steps at most.

Score calculation for a point is detailed below (Algorithm 7), and the functions involved can be seen in figure 2.4. Initially, a maximum distance is calculated to add 5 cm of additional distance with respect to the minimum distance. This just privileges additional distance, but points between the minimum and maximum distance are still valid, although they have a lower score. Then the gaussian component parameters (mean and standard deviation) are set up. These two parameters (standard deviation in particular) have been mainly chosen experimentally to spread evenly across the track without penalizing points at the edge of the track too much, while privileging the middle of the track, which is the same as saying that steps of mid length are privileged. After calculating the gaussian component on the Y axis thanks to the notorious gaussian probability density function (**gaussianDensity** in the algorithm), the algorithm evaluates obstacle points on the track to understand which one is the nearest to the evaluated point. Based on this distance value, the initial score is computed through a truncated linear equation that gives a score of 0 if an obstacle is closer than the minimum distance, 1 if further than the maximum distance, and between 0

and 1 if between minimum and maximum distance. A careful reader might notice that there's a huge gap between a point at minimum distance and a point slightly closer to an obstacle. The reason is that a smoother function should give a very close to zero in the first instance, and zero in the second. But while the second point has to be discarded, the first has to be taken into account, and having a value too close to zero would discard it too in practice. That's why the actual function has a big gap that allows point at minimum distance to be kept in consideration when calculating the next step. The gaussian component is then multiplied to the initial score to obtain the final score.

Algorithm 7 calculateScore

```
Input: point, obstaclesShape[type], userP, minDist[type]
Output: tracks, obstaclesShape
maxDist ← minDist[type] + 5 cm
mean ← userP.maxStepLength/2
stdDev ← 30 cm
gaussianComp ← gaussianDensity(point.y, mean, stdDev)
nearestObs ← +∞
for each obsPoint in obstaclesShape[type] do
  if ‖ obsPoint.y - point.y ‖ < maxDist then
    nearestObs ← ‖ obsPoint.y - point.y ‖
  end if
end for
if nearestObs < minDist[type] then
  score ← 0
else if nearestObs > maxDist then
  score ← 1
else
  score ← nearestObs / maxDist
end if
score ← score · gaussianComp
return score
```

The last function of the Computer Vision module is the one that will

Algorithm 8 tracksEvaluation

```
Input: plane, obstacles, userP
Output: tracks, obstaclesShape
maxObstacleHeight ← { 0, 0 }
for each point of obstacles do
  type ← insideTracks(point, userP)
  if type ≠ -1 then
    if obstaclesShape[type][point.y] doesn't exists
    or obstaclesShape[type][point.y] < point.z then
      obstaclesShape[type][point.y] ← point.z
      if maxObstacleHeight[type] < point.z then
        maxObstacleHeight[type] ← point.z
      end if
    end if
  end if
end for
if maxObstacleHeight[0] and maxObstacleHeight[1] < 20 cm then
  minDist ← setMinDist(maxObstacleHeight, userP)
  for each point of plane do
    type ← insideTracks(point)
    if type ≠ -1 then
      score ← calculateScore(point, obstaclesShape[type], userP,
      minDist[type])
      tracks[type] ← tracks[type] + {point,score}
    end if
  end for
end if
return tracks, obstaclesShape
```

output the optimal step length for each track. The algorithm is detailed below (Algorithm 9). The actual implementation relies on a sliding window (based on foot dimensions) calculation that will calculate the mean score of the points inside it. If a point inside a window has score equal to zero (which means that the point is closer than the minimum obstacle distance) the whole window will be discarded. The step length returned as output will be shifted so that the returned position will correspond to the centroid of the foot calculated on the Y axis.

2.3.7 Minimum Distance Parameter

Based on human walking bio-mechanics literature [63, 64], a minimum distance function has been developed to ensure that the selected footholds don't result in collisions with obstacles when executing the steps, since it has been demonstrated that the distance between foot and obstacles that ensures the avoidance of collisions depends on the height of such obstacle. This constraint can be formulated by the following equation with respect to the centroid of the foot:

$$minDist_c = maxObstacleHeight + safeDist \tag{2.9}$$

and, by translating the equation so that it relates to the edges of the foot (heel or tiptoe), the equation becomes:

$$minDist_c = maxObstacleHeight + safeDist - \frac{footLength}{2} \tag{2.10}$$

Parameter $safeDist$ has been empirically set at 10 cm , but can be modified to bring the feet closer to obstacles, for example in the case of environments cluttered by obstacles, where very precise feet positioning is required

to ensure the traversability. This parameter is the one that has been saved inside `userP` object (section 2.3.2.

2.4 Collision-Free Foot Trajectory Generator

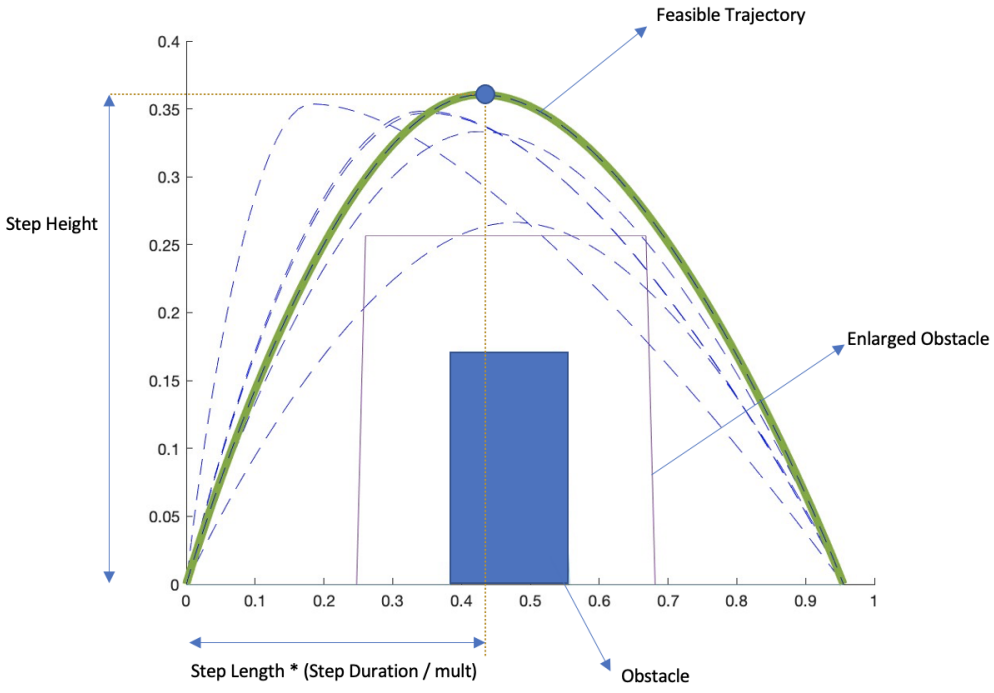


Figure 2.5 Example of CFFTG iterations

After obtaining obstacles' shape and step length, the Collision-Free Foot Trajectory Generator (CFFTG) has the goal of finding a feasible foot trajectory that avoids collisions with the detected obstacle points. Since the `obstacles_shape` object contains a function in the Y-Z plane $z = f(y)$, CFFTG has to find another function $z = g(y)$ representing the motion of the

Algorithm 9 footHoldEvaluation

```
Input: tracks, userP
Output: stepLength
windowPosition ← 0
stepLength ← { { }, { } }
bestScore ← { 0, 0 }
while windowPosition <= userP.maxStepLength - userP.footLength do
  for each of the two tracks do
    valid ← true
    tempScore ← 0
    nPoints ← 0
    while valid = true do
      for each index j of track i that satisfies:
      windowPosition < tracks[i][j].point.y < windowPosition +
      userP.footLength do
        if tracks[i][j].score ≠ 0 then
          tempScore ← tempScore + tracks[i][j].score
          nPoints ← nPoints + 1
        else
          tempScore ← 0
          valid ← false
        end if
      end for
    end while
    if nPoints ≠ 0 then
      tempScore ← tempScore / nPoints
    end if
    if tempScore > bestScore[i] then
      bestScore[i] ← tempScore
      stepLength[i] ← windowPosition + userP.footLength / 2
    end if
  end for
  windowPosition ← windowPosition + 1 cm
end while
return stepLength
```

centroid of the foot, such that

$$g(y) > f(y) \qquad \forall y \neq 0 \tag{2.11}$$

which amounts to say that the the centroid of the foot will pass over the obstacle. Also, $g(y)$ has to respect some constraints:

- Z value cannot exceed the maximum step height. In this thesis the value has been set to 40 cm;
- Since function $g(y)$ represents the motion of the centroid of the foot, it has to keep some distance with respect to the obstacle points.

To solve the aforementioned problem, I have employed a cubic polynomial function to describe the foot motion trajectory. To find the polynomial coefficients, at least three points are needed, but in the case of robots' kinematics, is not enough. In the case of the actual implementation, the function **cubicPolyTraj** provided by MATLAB also needs the time at which those points are reached, in order to return velocities, acceleration and positions for the entire foot motion during the step. After having understood that three Y-Z points and the related timestamps are needed, the problem can be formulated as follows: two points (the initial position of the foot, and the final position returned by the Computer Vision Module) and two timestamps (the initial timestamp is zero, the final depends on step duration and can be heuristically chosen within safe and reasonable limits, namely $finalTimeStamp = initialTimeStamp + stepDuration$) are already available. This is already enough to compute the Y-axis positions, velocities and accelerations, since in that axis the motion can be uniform without loss of generality. The foot trajectory algorithm will then provide the mid-swing

point Z coordinate (which amounts to the maximum step height) and timestamp. In practice, the value of the timestamp will affect the time in which the foot reaches the maximum height during the step. The value of the middle timestamp can then be written as:

$$middleTimeStamp = \frac{finalTimeStamp - initialTimeStamp}{mult} \tag{2.12}$$

so that the value of $middleTimeStamp$ only depends on $mult$ (remember that initial and final timestamps can be considered as given).
After having assessed what values are needed, we have to formulate the aforementioned constraints. The first one, related to height, is easily defined as:

$$stepHeight < maxStepHeight \tag{2.13}$$

The second one requires more reasoning, since it is important to model the physics behind collisions in a simple yet comprehensive way, to allow the algorithm to perform correctly in a reasonable time. A very well-known and simple obstacle avoidance method used in robotics is obstacle enlargement [65], which consist of increasing the dimensions of the detected obstacles based on the shape of the robot. In this way, we can plan robot's movement thinking of it as a point (usually the point coincides with its center of mass) [66] , and while the point doesn't collide with the enlarged obstacles, the actual robot won't collide with the real obstacles. It's important to note that each dimension of the obstacle will be enlarged by a different value in general, which depend on the robot's shape and the poses that it can assume. The values of enlargement for this thesis have been decided based on [66], which tracks the foot pitch angle of healthy subjects while walking. The highest absolute angle value tracked in the paper's experiments has been

about 50°, but it has to be considered that in these experiment the ankle was not restricted in any way, while the Lower Limb Exoskeleton that's assumed to be used in this thesis has no Degrees of Freedom on the ankle. That's why it has been chosen to assume a foot's maximum tilt angle of 45° during its motion. This reasoning implies that we can compute obstacles' enlargement in the following way:

- At 0° of tilt, the foot extends from its centroid in both directions of the Y-axis of $\frac{footLength}{2}$ cm (based on the geometric definition of centroid). Therefore obstacles have to be enlarged of $\frac{footLength}{2}$ cm on the Y-axis;
- At 45° of tilt, the foot extends from its centroid in decreasing direction of the Z-axis (namely towards the ground) of $\frac{footLength \cdot \sin(45°)}{2}$ cm. Therefore obstacles have to be enlarged of $\frac{footLength \cdot \sin(45°)}{2}$ cm on the Z-axis;

Let $f_2(y)$ the enlarged obstacles' function. Thanks to obstacle enlargement, now the only constraints needed to solve the problem are:

$$g(y) > f_2(y) \qquad \forall y \neq 0 \tag{2.14}$$

$$g(y) \leq maxStepHeight \qquad \forall y \tag{2.15}$$

As it can be deduced from the previous reasoning, the problem amounts to finding two values, $mult$ and $stepHeight$, that will make possible to obtain the Z-coordinate and timestamp needed to calculate positions, velocities and accelerations on the Z-axis.
The method chosen to solve the problem is a randomized iterative search [67]. The algorithm can be summarised as follows:

1. $mult$ and $stepHeight$ are drawn from two different gaussian random

variables with initial means and standard deviations. Initial mean for *mult* is set as 2, which represents a step which reaches the maximum height at $\frac{stepLength}{2}$. This values ensures that the first iteration has equal probability of "moving" the curve to the left and to the right. Initial mean for $stepHeight$ is set as $maxObsHeight + \frac{footLength \cdot \sin(45°)}{2}$, which represents the minimum distance that the centroid has to keep to ensure no collisions on the Z-axis. Initial standard deviations are set as 2 for *mult* and 0.03 cm for $stepHeight$, and have been found experimentally to return a small number of iterations;

2. The trajectory for such values is computed and a score is given to the trajectory based on the minimum euclidean distance between points of $g(y)$ and $f_2(y)$;

3. Point **2** is iterated again and new *mult*, $stepHeight$ and score are computed;

4. If the score is higher than the previous one, the new values of *mult* and $stepHeight$ become the updated means of the two gaussians random variables, and the standard deviations are reduced by a scaling factor. The scaling factor has been experimentally set as 2 for *mult* and 1.5 for $stepHeight$;

5. Point **3** is repeated until a trajectory score ≥ 0 is found, or the maimum number of iteration is reached in case of failure.

The pseudocode for the proposed CFFTG is described in Algorithm 10.

Algorithm 10 collisionFreeFootTrajectoryGenerator

Input: obstaclesShape, maxObsHeight, userP, stepLength, varianceMult, varianceHeight, initialTimeStamp

Output: trajectory

```
safeDist ← (userP.footLength / 2) + 1 cm
mult ← 2
bestMult ← mult
stepHeight ← maxObsHeight + safeDist · sin (45)
bestHeight ← stepHeight
enlargeObstacles(safeDist)
score ← 0
bestScore← −∞
while score ≤ 0 do
  score ← +∞
  waypointsY ← {0, stepLength }
  timeStampsY ← {0, userP.stepDuration }
  middleTimeStamp ← userP.stepDuration / mult
  middleZ ← stepHeight
  waypointsZ ← {0, middleZ, 0 }
  timeStampsZ ← {initialTimeStamp, middleTimeStamp, initialTimeS-
  tamp + userP.stepDuration}
  trajectory ← cubicPolyTraj(waypointsY, timeStampsY, waypointsZ,
  timeStampsZ)
  for each tPoint of trajectory do
    oPoint ← obstaclesShape[tPoint.y]
    if tPoint.z ≤ oPoint.z then
      tempScore ← - ‖ (tPoint - oPoint) ‖
    else
      tempScore ← + ‖ (tPoint - oPoint) ‖
    end if
    if tempScore < score then
      score ← tempScore
    end if
  end for
  if bestScore < score then
    bestScore ← score
    bestMult ← mult
    bestHeight ← stepHeight
    varianceMult ← varianceMult / 2
    varianceHeight ← varianceHeight / 2
  end if
  mult ← randomGaussian(bestMult, varianceMult)
  stepHeight ← randomGaussian(bestheight, varianceHeight)
end while
return trajectory
```

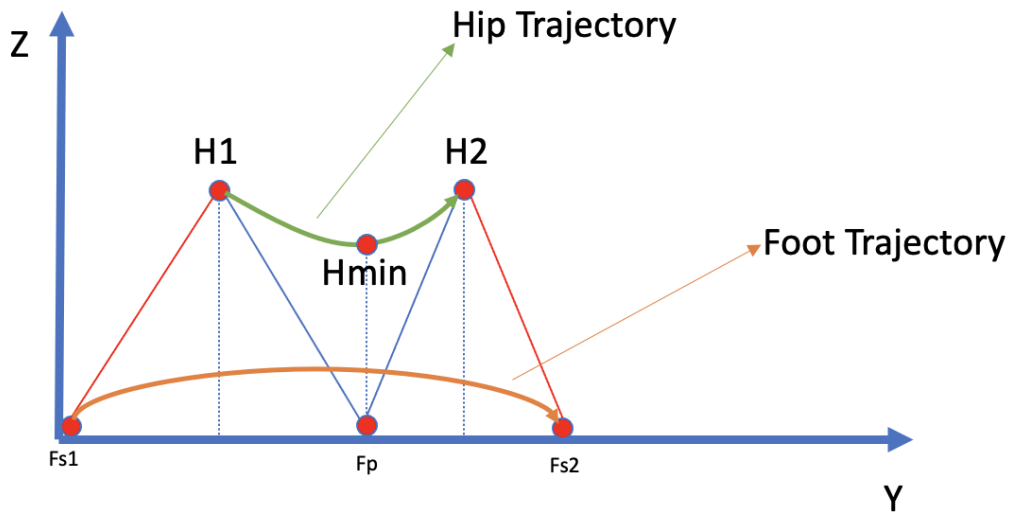


Figure 2.6 Simplified scheme of the kinematic model.
The swing leg is depicted in red, the support leg is depicted in blue.

2.5 Exoskeleton Kinematic Model

After finding an appropriate trajectory for the foot, a kinematic model is needed to compute the joint angles that will be then fed to the PIDs to produce the actual motion of the exoskeleton. A simplified scheme of the chosen kinematic model is shown in figure 2.6:

- F_{s1} and F_{s2} represent the swing foot's centroid initial and final positions during a step; the orange arrow represents the trajectory computed in the previous section;

- H_1 and H_2 represent hip initial and final positions during a step; the green arrow represents the hip trajectory;

- F_p represents the position of the pivot foot's heel, which is related to

the leg that will stay in support during the swing.

After modeling an appropriate trajectory for the hip, the only unknown quantities needed to calculate the joint angles of this model are the position of the knees for swing and pivot legs, K_p and K_s, and the position of the heel F_h with respect to the centroid of the swing foot F_s; all of these quantities can be calculated using trigonometry.

To model hip trajectory, the following assumptions have been made:

- Thigh and shank length, L_1 and L_2 are known parameters;

- $H_1.z = H_2.z = H.z$ can be fixed for each different user, based on the hip height that make the walk most comfortable for them. The only constraints that must be kept in mind are:

$$H.z \leq L_1 + L_2 \tag{2.16}$$

$$H.z \geq 10\%(L_1 + L_2) \tag{2.17}$$

  The first condition represents the fact that the hip can't reach an height higher than the fully extended leg. The second represents the intuitive idea that, if the hip is too close to the ground, the knee will excessively bend, that does not represent a physiological walking;

- $H_1.y$ and $H_2.y$ can be found by calculating the mid point between the two feet. Since in double support heel and centroid of the foot lies on the Y-axis, the two values can be computed as:

$$H_1.y = \frac{F_p - (F_{s1} - \frac{footLength}{2})}{2} \tag{2.18}$$

$$H_2.y = \frac{(F_{s2} - \frac{footLength}{2}) - F_p}{2} \tag{2.19}$$

- The trajectory will be modeled as a cubic polynomial that reaches a minimum when $y = F_p$ [68];
- The minimum mentioned above will be found using the piston motion equations [69].

Since it has been chosen to model hip trajectory with a cubic polynomial, we need timestamps and waypoints to compute the trajectory, as it has been done in section 10. Since we already have H_1 and H_2 Y-axis and Z-axis coordinates, the trajectory on the Y-axis can be computed by modeling it as uniform, as previously done in section 10, with $timeStamp(H_1) = initialTimeStamp$ and $timeStamp(H_2) = initialTimeStamp + stepDuration$. we just need to find the timestamp $timeStamp(H_{min})$, at which the minimum of the hip trajectory occurs, and the Z coordinate of H_{min}, to calculate the trajectory on the Z-axis.

The percentage of the step duration at which the minimum occurs is then given by:

$$step\% = \frac{F_p - H_1.y}{H_2.y - H_1.y} \tag{2.20}$$

Then, the time at which the minimum occurs will be:

$$timeStamp(H_{min}) = step\% \cdot (H_2.y - H_1.y) \tag{2.21}$$

To find the Z coordinate of H_{min} the piston motion equations can be used, in particular the ones related to the crank connecting rod mechanism. Figure 2.7 shows the time at which the minimum occurs, which happens to be when the hip Y-axis position and the pivot foot Y-axis position have the

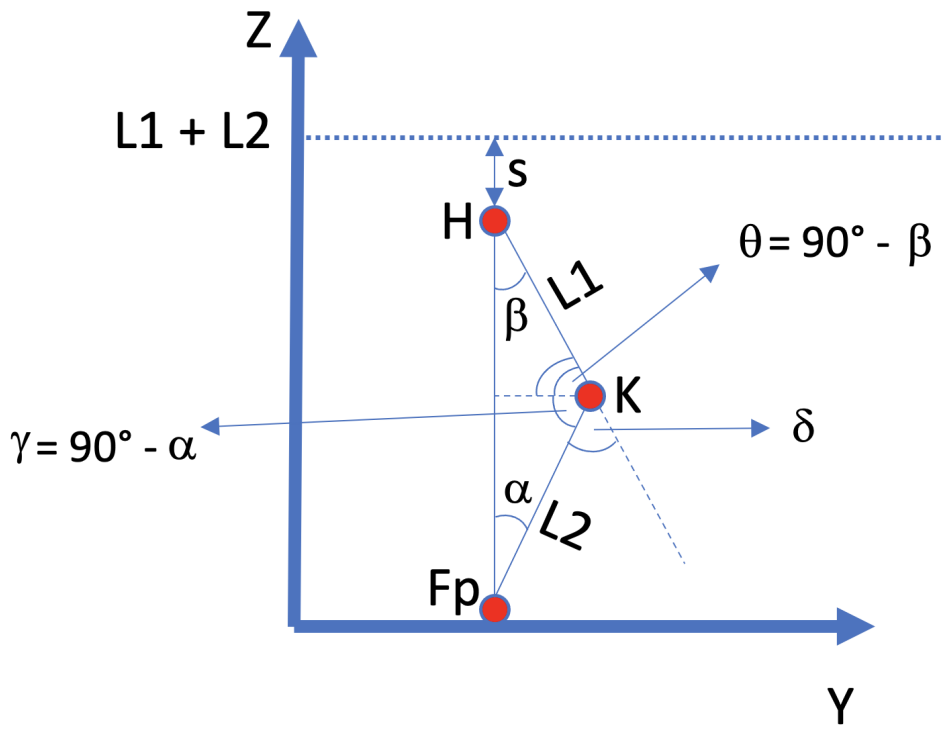


Figure 2.7 Crank Connecting Rod Scheme for support leg

same value.

From Crank Connecting Rod equation [69], it can be showed that:

$$s = L_1 + L_2 - (L_1 \cdot \cos(\beta) + L_2 \cdot \cos(\alpha)) \tag{2.22}$$

δ, which is the knee bending angle, will be considered as given, and the ultimate goal is to find the value of s, which will be subtracted to $H.z$ to obtain $H_{min}.z$.

Figure 2.7 also highlights some useful relationships:

$$\delta = 180° - \theta - \gamma = \alpha + \beta \tag{2.23}$$

$$L_2 \sin\alpha = L_1 \sin\beta \rightarrow \beta = \arcsin(\frac{L_2}{L_1} \cdot \sin(\alpha)) \tag{2.24}$$

Then we can find α with respect to β:

$$\alpha = \delta - \beta = \delta - \arcsin(\frac{L_2}{L_1} \cdot \sin(\alpha)) \tag{2.25}$$

$$\arcsin(\frac{L_2}{L_1} \cdot \sin(\alpha)) = \delta - \alpha \tag{2.26}$$

$$\frac{L_2}{L_1} \cdot \sin(\alpha) = \sin(\delta - \alpha) = \sin\delta \cdot \cos(\alpha) - \cos(\delta) \cdot \sin\alpha \tag{2.27}$$

$$(\cos(\delta) + \frac{L_2}{L_1}) \cdot \sin(\alpha) = \sin\delta \cdot \cos(\alpha) \tag{2.28}$$

$$\tan(\alpha) = \frac{\sin(\alpha)}{\cos(\alpha)} = \frac{\sin(\delta)}{(\cos(\delta) + \frac{L_2}{L_1})} \tag{2.29}$$

$$\alpha = \arctan(\frac{\sin(\delta)}{\cos(\delta) + \frac{L_2}{L_1}}) \tag{2.30}$$

At this point all quantities are defined (remember that δ can be fixed) and s can be computed using 2.22. In this work $\delta = 20°$ has been chosen, although a different value might be used after assessing the most comfortable knee bending for the user.

After defining hip trajectory, we can proceed to the actual inverse kinematics computation of the leg angles that will be then fed into the PIDs. It's

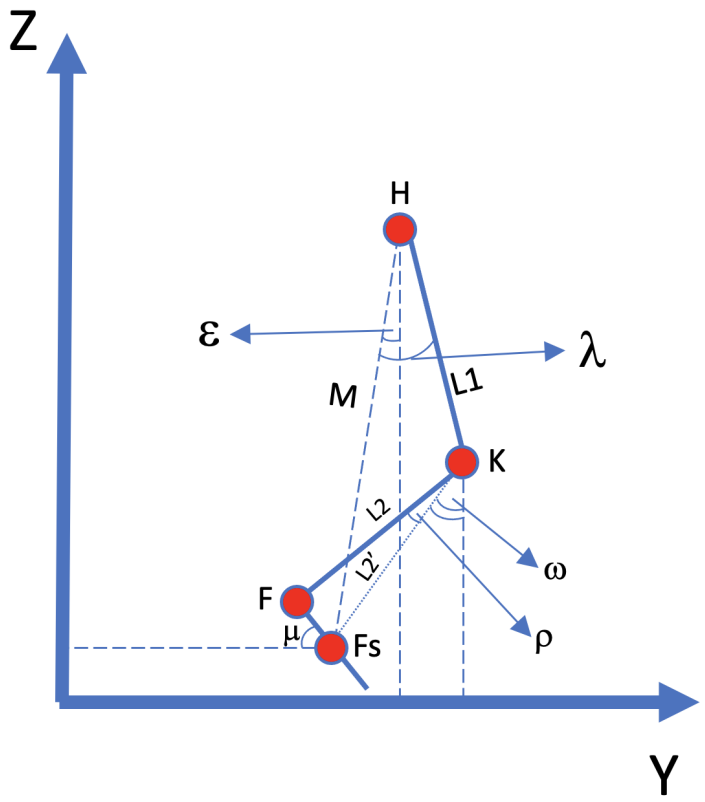


Figure 2.8 Scheme used to calculate knee and heel position

worth mentioning that heel position of the support leg F_p is known, since the foot completely lies on the Y-axis, but generally the position of the heel F for the swing leg isn't known, and is necessary to obtain the correct angles. The goal is to find positions H (hip) , K (knee) and F (heel) for each leg. Then the angles will be obtained as [57]:

$$\theta_H = \arctan(\frac{H.z - K.z}{K.y - H.y}) \tag{2.31}$$

$$\theta_K = \theta_H - \arctan(\frac{K.z - F.z}{F.y - K.y}) \tag{2.32}$$

(Note that, for the support leg, $F = F_p$).

The procedure below will compute K for the swing leg, to obtain K for the support leg is sufficient to use F_p instead of F_s and L_2 instead of L'_2. To compute K with respect to H and foot centroid F_s, we first compute L'_2, which amounts to:

$$L'_2 = \sqrt{L_2^2 + \|(F_s - F)\|^2} = \sqrt{L_2^2 + (\frac{footLength}{2})^2} \tag{2.33}$$

since the distance between heel and centroid of the foot is $\frac{footLength}{2}$ by definition of centroid.

After computing $M = \|(F_s - H)\|$, we can use the Cosine Rule to find λ:

$$\cos(\lambda) = \frac{M^2 + L_1^2 - L_2^2}{2ML_1} \tag{2.34}$$

$$\lambda = \arccos(\frac{M^2 + L_1^2 - L_2^2}{2ML_1}) \tag{2.35}$$

We also need ε which represents the rotation of the triangle $\triangle HKF_s$:

$$\varepsilon = \arcsin(\frac{F_s.y - H.y}{M}) \tag{2.36}$$

At this point position K can be computed as:

$$K.y = H.y + L_1 \sin(\lambda + \varepsilon) \tag{2.37}$$

$$K.z = H.z - L_1 \cos(\lambda + \varepsilon) \tag{2.38}$$

At this point, every quantity needed to compute angles for the support leg has been calculated. An additional computation is needed for the swing leg, since we don't have the heel position F. To find it, a similar procedure with respect to the computation of K can be exploited. Note that the following

computation will work under the simplified assumption of a fixed / passive joint for the ankle that keeps the angle between the shank and the foot at 90°, at least when the leg is swinging.

First, angles ω and ρ can be computed as:

$$\omega = \arcsin(\frac{F_s.y - K.y}{L'_2}) \tag{2.39}$$

$$\rho = \arcsin(\frac{\frac{footLength}{2}}{L'_2}) \tag{2.40}$$

Finally, after assessing that $\mu = \rho - \omega$ position F can be computed as:

$$F.y = F_s.y - \frac{footLength}{2} \cdot cos(\mu) \tag{2.41}$$

$$F.z = F_s.z + \frac{footLength}{2} \cdot sin(\mu) \tag{2.42}$$

At this point the computed values can be fed to 2.31 and 2.32 to obtain the angles.

3 | Experiments and Results

3.1 Experimental Setup

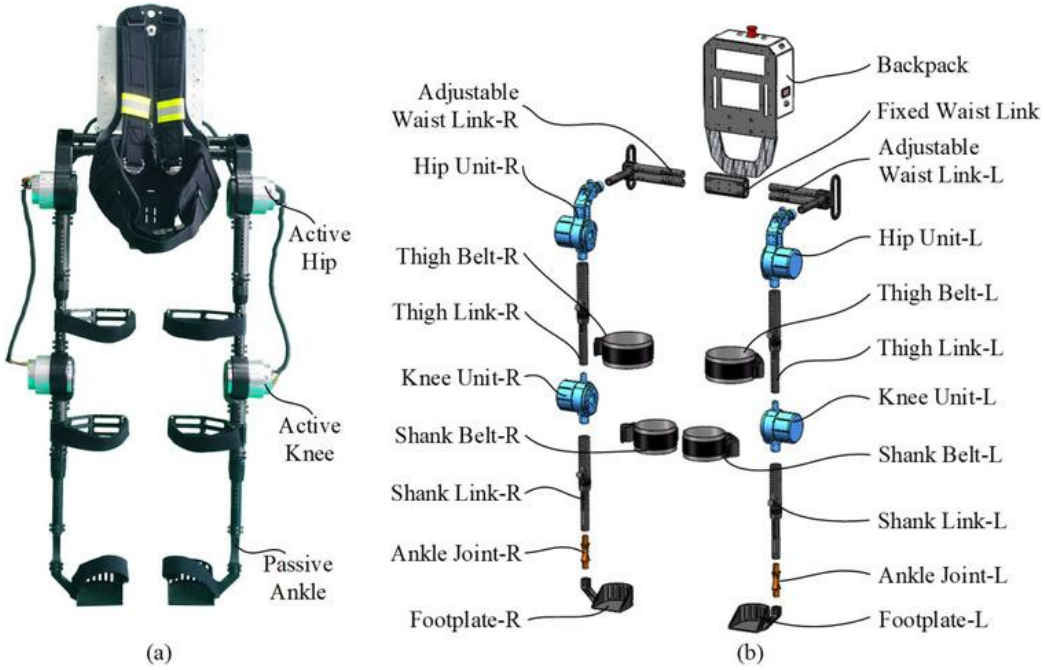


Figure 3.1 Simplest type of LLE [5]

The experiments to validate the proposed model are based on the acquisition of different real-world scenarios (i.e., obstacles of different shape and positions) and on the evaluation of the exoskeleton kinematics in these

conditions. To sense the environment and detect the obstacles, an Intel® RealSense™ D455 stereo depth camera was adopted and supposed to be rigidly fixed at the exoskeleton pelvis. Detailed information, including camera's datasheet, are available at [70]. Figure 3.2 shows the experimental setup.

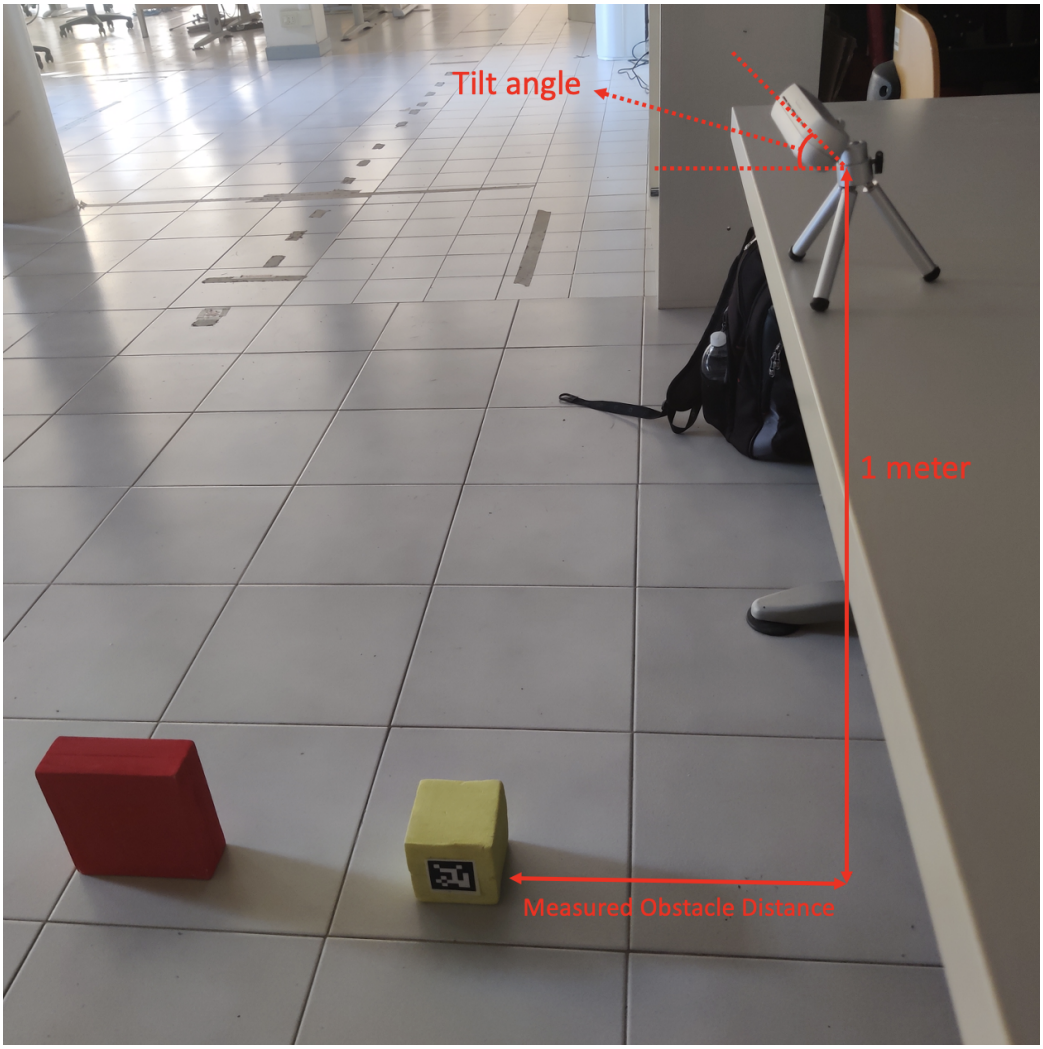


Figure 3.2 Setup employed for the experiments

This type of camera relies on Time of Flight and Stereo technologies, and can produce high quality 3D point clouds that can then be processed to

extract huge amount of visual information. The reader can take a look at [71] and [72] to better understand these technologies. The reason why this component has been chosen is mainly the precision of the sensor (less than 2% of depth error at 4 meters of distance) and the high acquisition rate (namely 90 FPS), which is fundamental when dealing with real-time tasks, such as locomotion. Furthermore, since this camera measures infrared radiation but doesn't always project it (it uses any light to measure depth, and project it only if necessary), it can work perfectly both indoor and outdoor, and with negligible cross-talk with other similar devices. It is also worth noting that the D455 camera comes with an Inertial Measurement Unit (IMU), therefore providing additional information about the pose of the camera such as rotation, acceleration and magnetic field.
Another important aspect relates to the framework that will be used to create those software modules. The framework that has been chosen to implement the computer vision module is Robot Operating System (ROS), an open-source framework that helps researchers and developers build and reuse code between robotics applications. This framework has many advantages, to name a few:

- Software modules come in form of packages, allowing cross-collaboration between developers without having to reinvent the wheel every time;
- Algorithms can be coded in many different languages, the main ones being C++ and Python;
- A simple communication structure based on processes (nodes) which can broadcast messages (topic publishers) or listen to them (topic subscribers). Ideally, there should be a node for each sub task that the robotic system has to complete in order to solve the whole task;

- Allows to simulate the task execution before deploying the solution on the real robotic system.

For more detailed information about ROS the reader is advised to have a look at [73] and [74]. The ROS distribution that was used in this work is Noetic.

Due to the unavailability of a fully functional exoskeleton to test the proposed method, all the experimental evaluation have been carried out on a virtual exoskeleton model implemented in MATLAB. The model consists of two thigh, shank and foot links, two hip and knee active joints, and two ankle passive joints that have been assumed fixed at a 90° angle with respect to the shank during the swing motion (the kinematics of a possible spring joint for the ankle has not been considered, since its relevance would mostly emerge when the feet is in contact with the ground, while this thesis is concerned with the behaviour of the foot during the swing motion). This results in a model with 4 active DoF and 6 joints, and it's use is validated by the fact that the majority of assistive lower limb exoskeletons are characterized by at least 6 joints, as shown in figure 3.1, 4 of which are generally actuated by DC motors (i.e., hip and knee pitch joints). Some LLEs also comprise active ankle joints (i.e., ankle dorsiflexion) and/or hip roll joints (i.e., leg abduction/adduction) for balancing against gravity. However, since they are not commonly adopted and I assume that balancing is maintained with crutches, or other passive tools, I didn't consider these additional joints in the exoskeleton model. The most common type of actuator for the active joints is by far the DC motor, as it could be noticed in table 1.1; since this work doesn't pose emphasis on the specific hardware actuation but it aims at generalizing to a variety of exoskeletons in the market or in research, I assumed that a standard model comprising a DC motor along with an encoder and a low-level PID controller

to control any active joint. The abstraction of a general bipedal robot with respect to an assistive Lower Limb Exoskeleton is based on the assumption the user's leg are completely passive and do not interfere with the exoskeleton dynamic.

3.2 Experiments

I made six experiments in different conditions (labeled from **(a)** to **(f)**), so that the robustness of the proposed method could be tested with different obstacle configurations. Since the algorithm is supposed to evaluate footholds for one foot at a time, the right leg has been chosen as the swing leg for all experiments without loss of generality. The position of the single-support (pivot) foot was supposed to be known (e.g., from the previous step and exoskeleton encoders) and it is assumed to be compliant with the constraint of the CV module (e.g., at least *minDist* from surrounding obstacles).
Experiment **(a)** represents one of the most common situations, where the obstacle is one-sided with respect to the two legs, and the support leg is positioned quite ahead of the camera (the left leg has more freedom in this case, since no obstacle is present on that track).
Experiment **(b)** shows a slightly more challenging situation. In this case, the obstacle occupies both tracks, and the support foot can no longer be placed as arbitrarily as before.
Experiment **(c)** shows a situation where the obstacle is far from the subject's footholds.
Experiment **(d)** shows a situation where the swing foot initial position is at minimum safe distance with respect to the obstacle.
Experiment **(e)** depicts a very challenging situation, where two obstacle are

present on the same track, and the space between them is the minimum required to position the foot.
Experiment **(f)** depicts the robot in double support with feet at the same Y-axis position (symmetrical with respect to the X-axis), a situation that wasn't considered in the previous experiments.

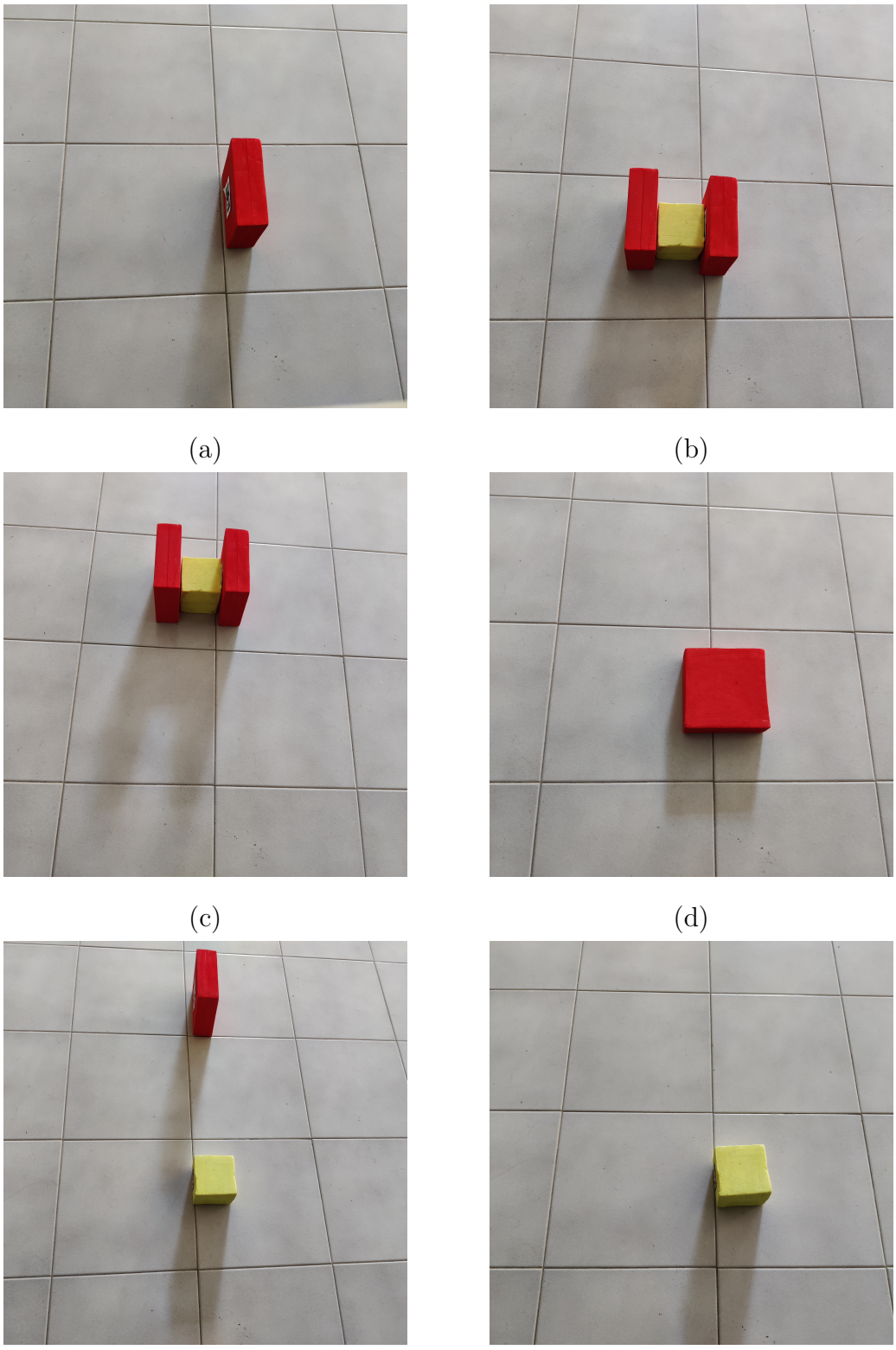
3.3 Evaluated Metrics

For each experiment, a table containing the notable measurements has been produced. The real measurements have been collected by using a measuring tape in the case of distances, and time measuring software tools such as the clock object in C++ and functions tic/toc in MATLAB; on the other hand, estimated measurements are those directly output by the CV and CFFTG modules. The table comprises:

- Geometric features of the detected obstacles (distance with respect to the camera, height and length);
- Support foot placement (which, together with the fact that the camera is assumed to be placed in the origin, allows to calculate initial swing foot position thanks to the kinematic model described in section 2.5);
- swing foot final placement (main output of the CV module);
- Total execution time of CV and CFFTG module;
- Values of $mult$ and $stepHeight$ for the trajectory output by the CFFTG module (explained in section 10);
- Number of iterations of the CFFTG module.

3.4 Results

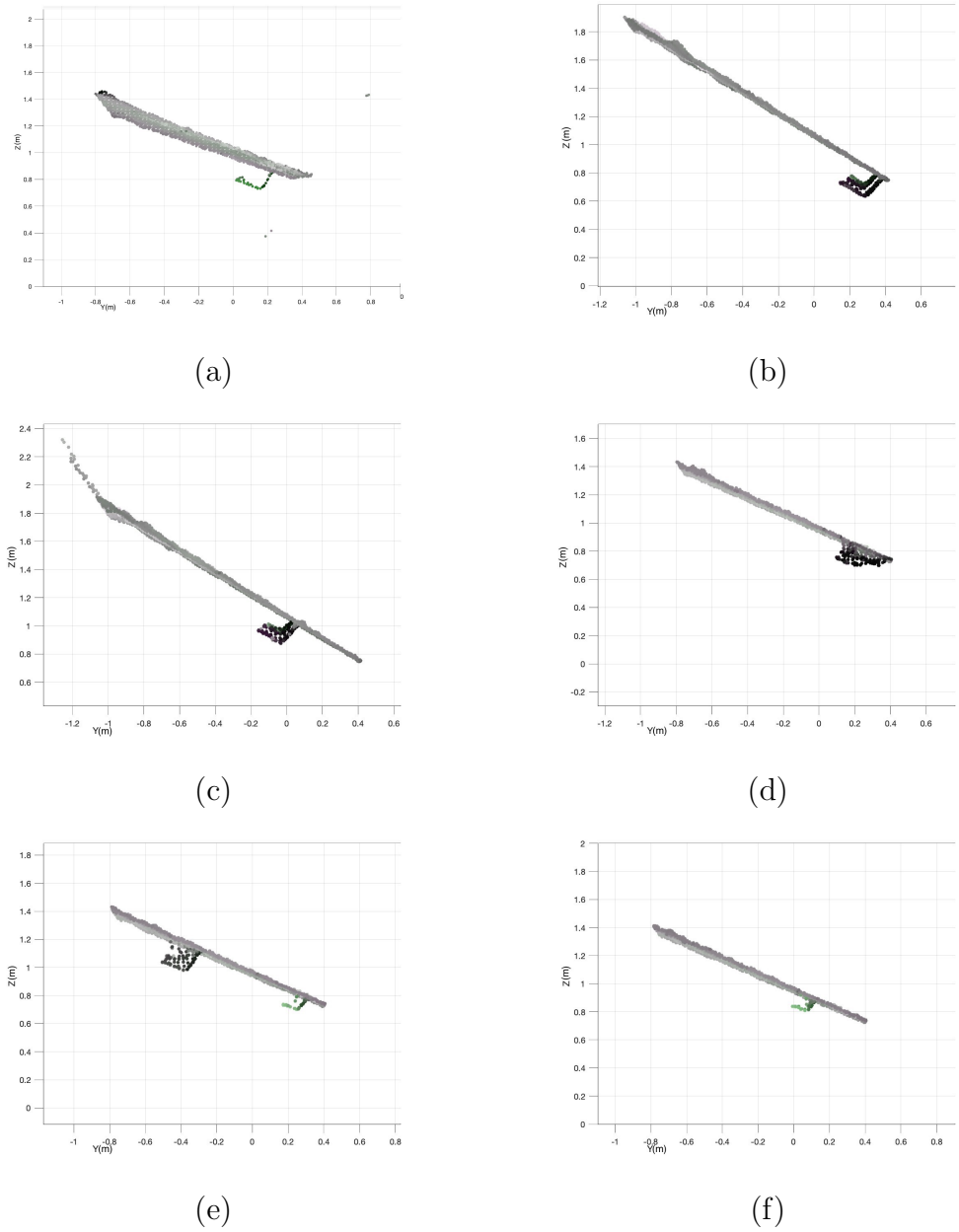
This section contains the experimental data collected to evaluate the performance of the whole pipeline, from filtered cloud to the simulated kinematics of the exoskeleton.
Figure 3.3 shows the RGB images of the experimental environments, seen from the camera's point of view.



(a) (b) (c) (d) (e) (f)

Figure 3.3 RGB Images of the experimental environments

Figure 3.4 to 3.8 shows the point clouds from the experiments, displayed with respect to the robot frame. Figure 3.4 shows the filtered point clouds of experiments **(a)** to **(f)** displayed in the sagittal plane, while figure 3.5 shows the filtered point clouds in the horizontal plane (bird-eye view).



(a) (b) (c) (d) (e) (f)

Figure 3.4 Filtered Clouds (Sagittal Plane)

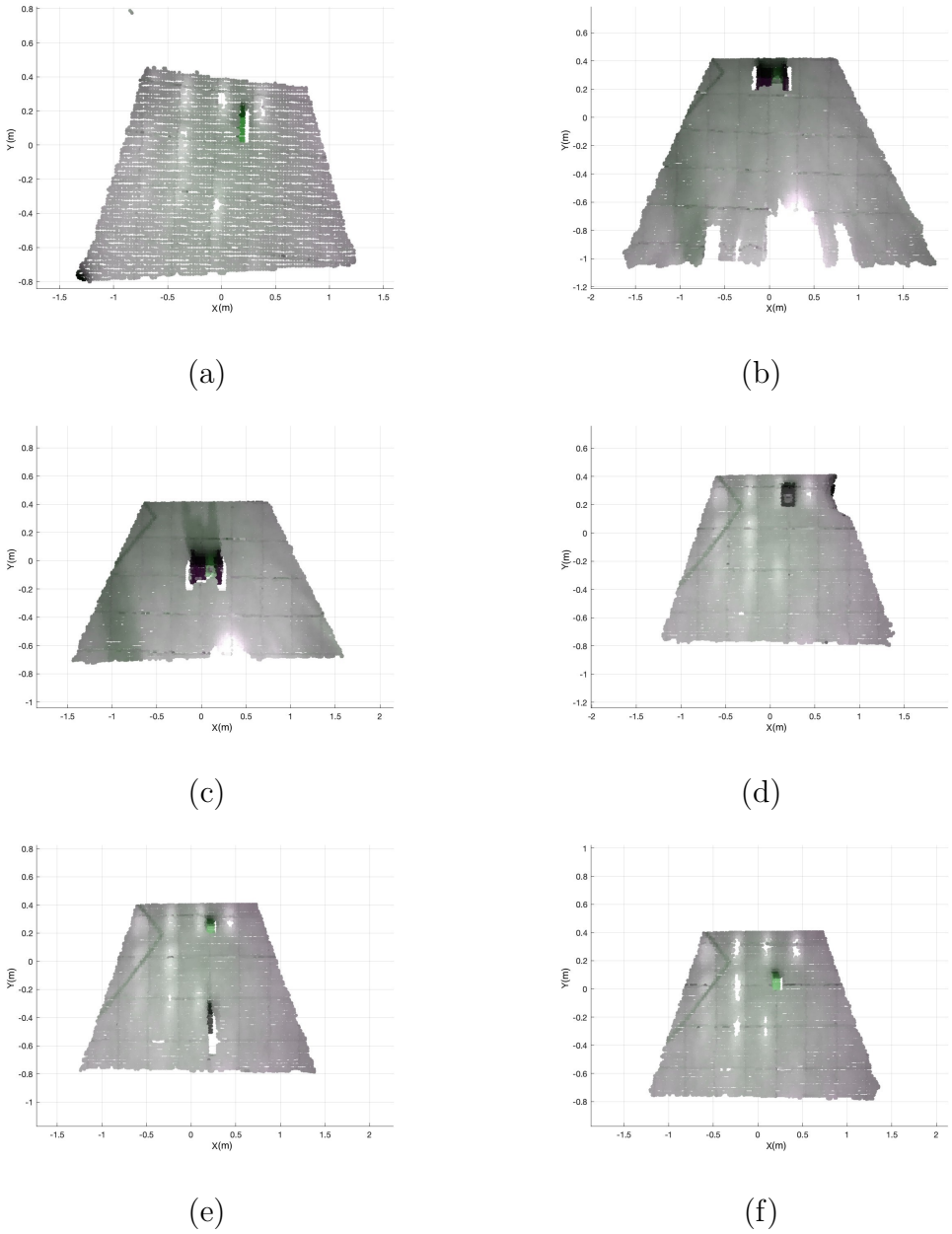


Figure 3.5 Filtered Clouds (Horizontal Plane)

Figure 3.6 shows the point clouds after ground plane detection and homogeneous transformation are applied, displayed in the sagittal plane of the

robot frame, while figure 3.7 shows the same point clouds in the horizontal plane; detected obstacle points are highlighted in green. It can be clearly seen that the CV module is able to correctly align the point cloud with the robot frame.

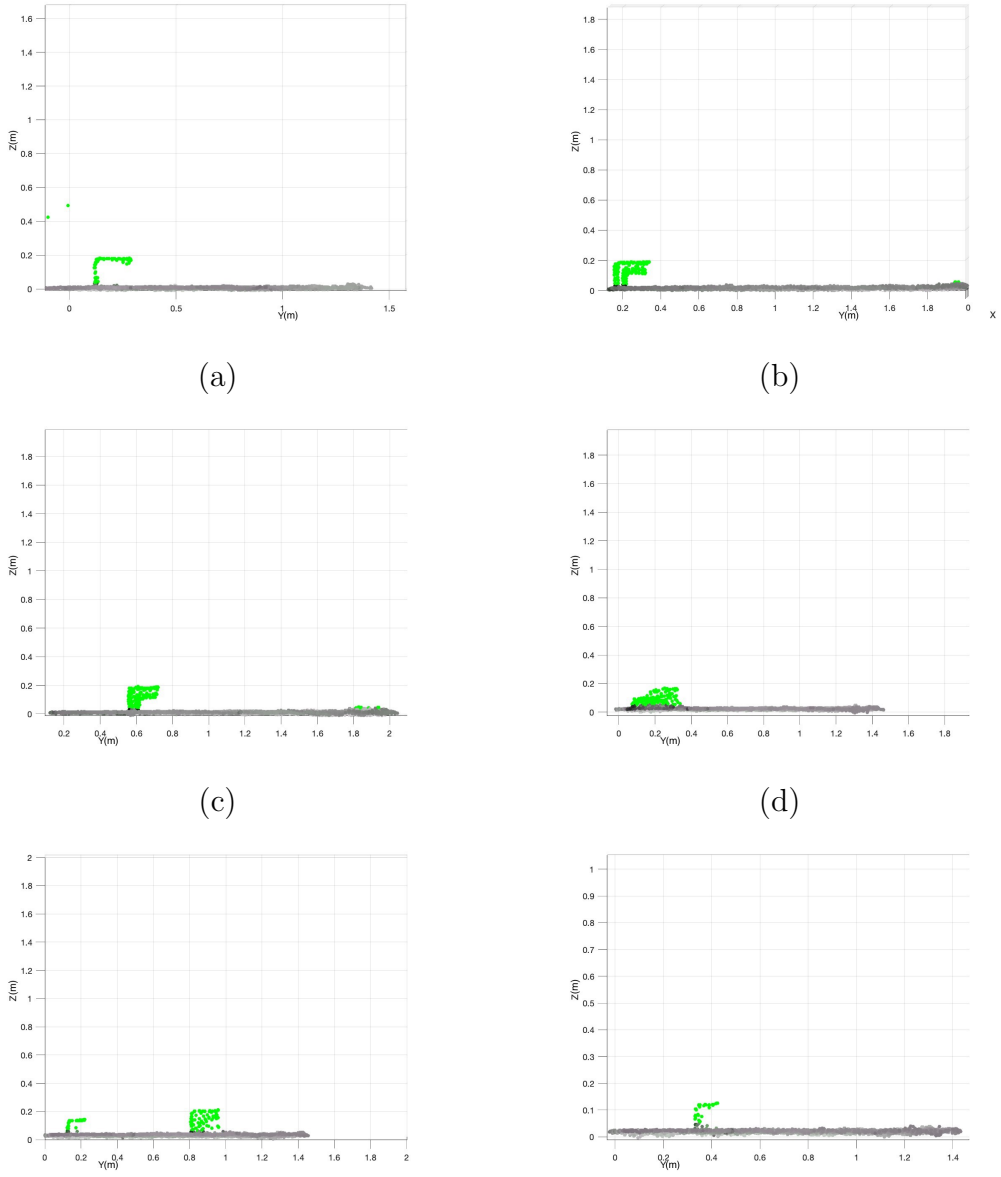


Figure 3.6 Clouds after Obstacle Detection and Homogeneous Transformation (Sagittal Plane)

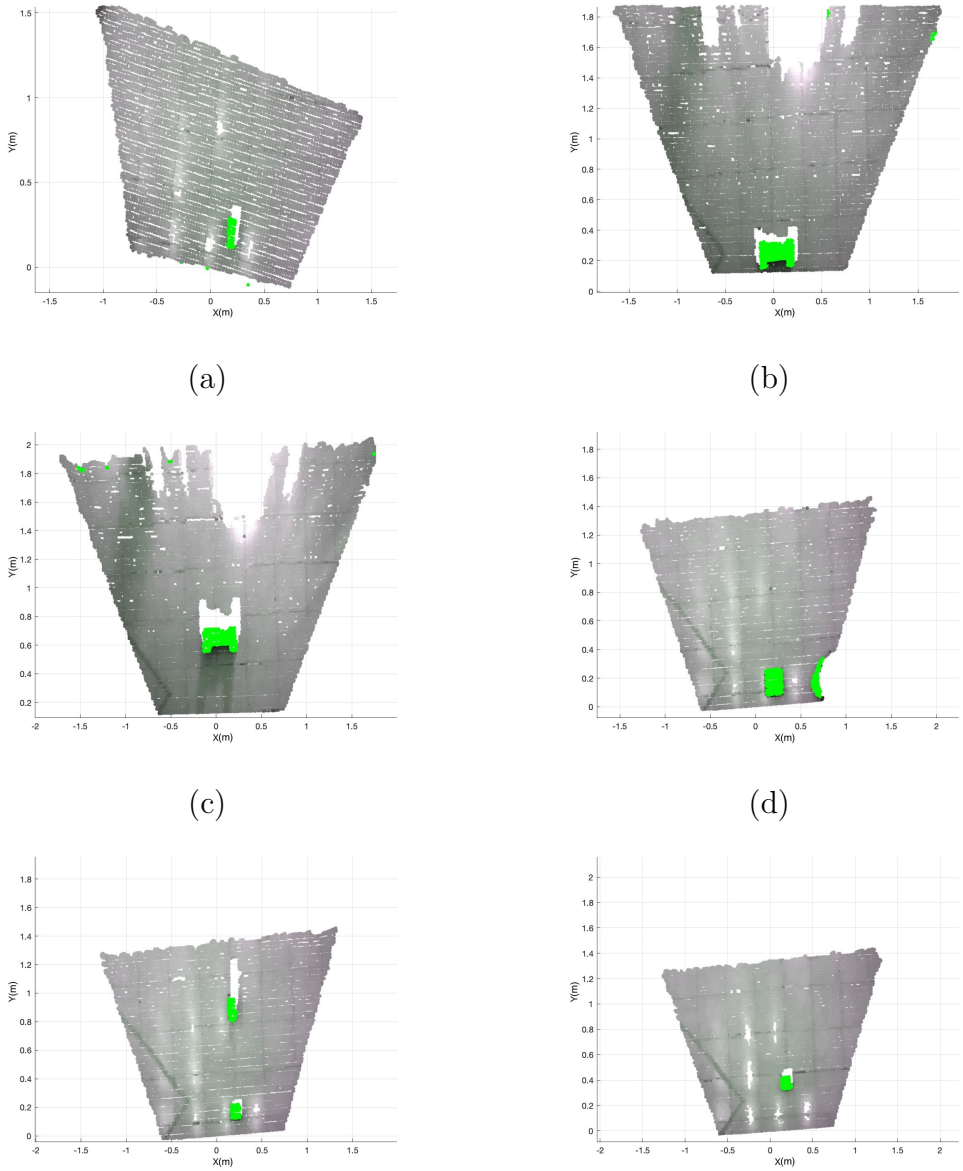


Figure 3.7 Clouds after Obstacle Detection and Homogeneous Transformation (Horizontal Plane)

Figure 3.8 shows the final point clouds after the whole CV is applied,

shown in the horizontal plane. In this case, it is sufficient to show the bird-eye view, since the sagittal view would be identical to figure 3.6. The two tracks representing feet's forward motion can be easily distinguished and have different colors, and the brightness of the tracks' points represents the probability of it being chosen for the foothold. The selected footholds, represented by squares inside the tracks, are highlighted in green. Notice that obstacle points inside the tracks are colored in black.

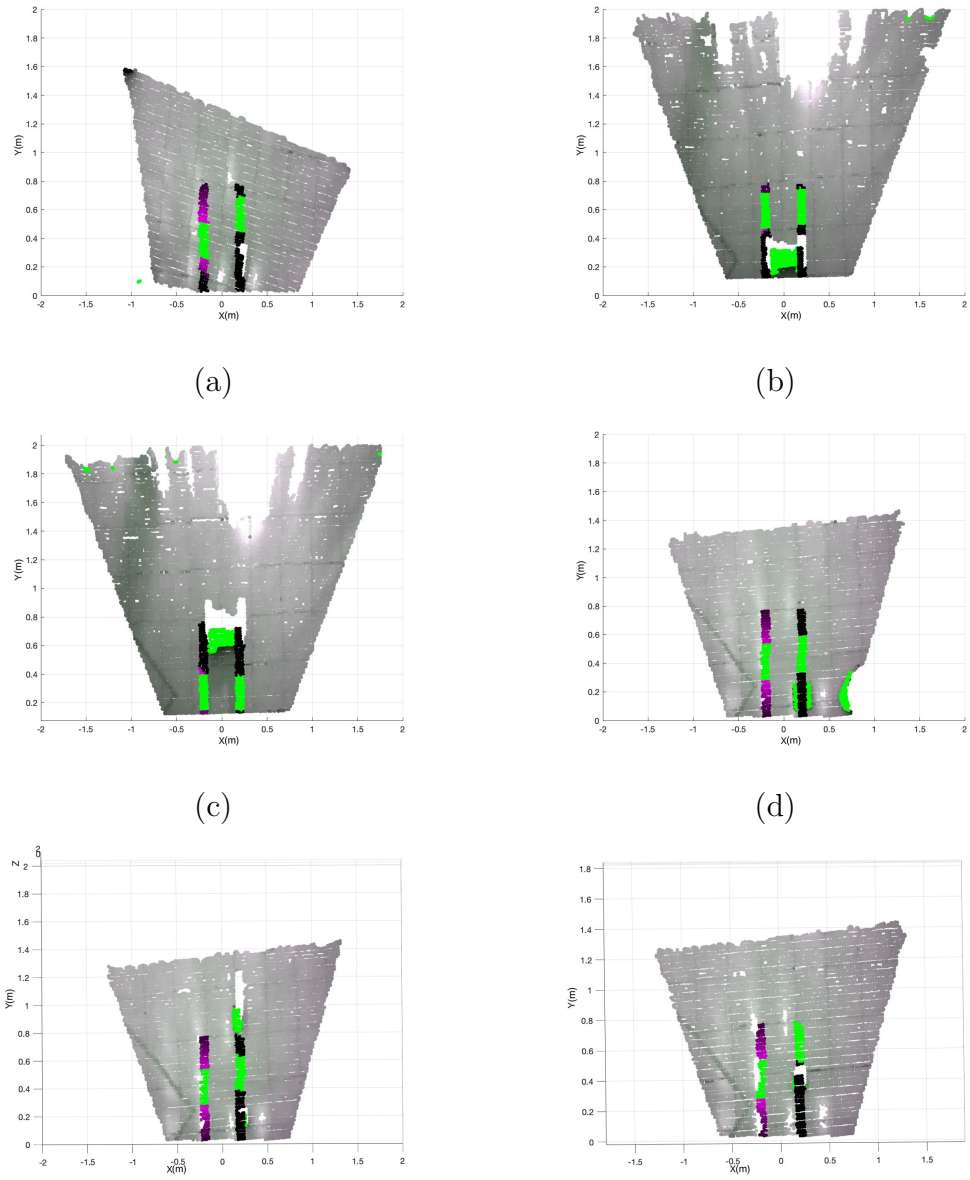


Figure 3.8 Clouds after the whole CV module is applied (Horizontal Plane).

Figure 3.9 represent the kinematic simulations comprising the swing (depicted in red) and support leg (depicted in blue), along with foot centroid

trajectory (depicted in light blue), hip trajectory (depicted in green) and detected obstacles. The reader might notice that the feet goes under the X-Y plane in some parts of the aforementioned figures. In a real situation, the passive ankle joint would change its angle so that the feet can fully lay on the ground, but since that condition doesn't affect the experiment, which is focused on the traversability of the obstacles, this behaviour has not been modeled.

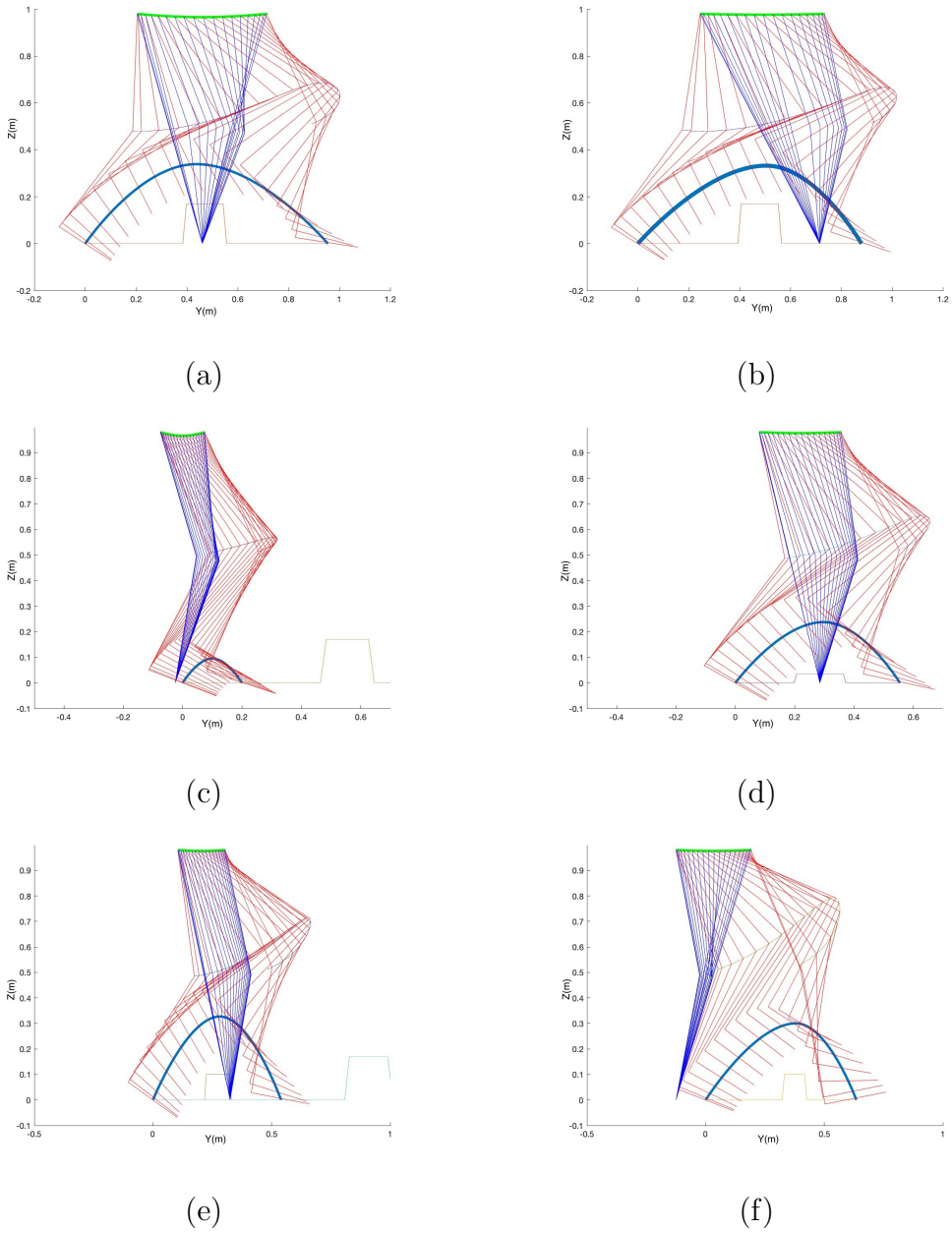


Figure 3.9 Visualization of the expected step kinematics. The support leg is depicted in blue, the swing leg is depicted in red. The Y-axis is shifted to place the swing's foot centroid in 0.

Tables 3.1 to 3.6 contains the evaluated metrics (detailed in the previous section) for each experiment.

Value	**Real Measure**	**Estimated Measure**
Obstacle 1 Distance	15 cm	15.57 cm
Obstacle 1 Length	17 cm	17.32 cm
Obstacle 1 Height	17 cm	17.23 cm
Support's Foot Placement (Heel)	-	23.5 cm
Swing's Foot Final Placement (Centroid)	-	62.5 cm
Total Execution Time (CV Module)	36.4 ms	-
Mult Value (CFFTG)	-	2.19
Step Height Value (CFFTG)	-	33.9 cm
Number of Iterations (CFFTG)	-	9
Total Execution Time (CFFTG)	90 ms	-

Table 3.1 Experimental Data (a)

Value	**Real Measure**	**Estimated Measure**
Obstacle 1 Distance	15 cm	15.3 cm
Obstacle 1 Length	17 cm	17.44 cm
Obstacle 1 Height	17 cm	17.16 cm
Support's Foot Placement (Heel)	-	47 cm
Swing's Foot Final Placement (Centroid)	-	63.5 cm
Total Execution Time (CV Module)	77.6 ms	-
Mult Value (CFFTG)	-	2.41
Step Height Value (CFFTG)	-	35.8 cm
Number of Iterations (CFFTG)	-	17
Total Execution Time (CFFTG)	150 ms	-

Table 3.2 Experimental Data (b)

Value	Real Measure	Estimated Measure
Obstacle 1 Distance	55 cm	55.2 cm
Obstacle 1 Length	17 cm	17.13 cm
Obstacle 1 Height	17 cm	17.48 cm
Support's Foot Placement (Heel)	-	5 cm
Swing's Foot Final Placement (Centroid)	-	27.5 cm
Total Execution Time (CV Module)	76.2 ms	-
Mult Value (CFFTG)	-	2
Step Height Value (CFFTG)	-	9.5 cm
Number of Iterations (CFFTG)	-	1
Total Execution Time (CFFTG)	100 ms	-

Table 3.3 Experimental Data (c)

Value	**Real Measure**	**Estimated Measure**
Obstacle 1 Distance	7.5 cm	7.66 cm
Obstacle 1 Length	17 cm	17.09 cm
Obstacle 1 Height	5 cm	3.88 cm
Support's Foot Placement (Heel)	-	16 cm
Swing's Foot Final Placement (Centroid)	-	51.5 cm
Total Execution Time (CV Module)	54 ms	-
Mult Value (CFFTG)	-	1.91
Step Height Value (CFFTG)	-	23.7 cm
Number of Iterations (CFFTG)	-	28
Total Execution Time (CFFTG)	340 ms	-

Table 3.4 Experimental Data (d)

Value	Real Measure	Estimated Measure
Obstacle 1 Distance	12 cm	12.16 cm
Obstacle 1 Length	10 cm	10.33 cm
Obstacle 1 Height	10 cm	9.87 cm
Obstacle 2 Distance	79 cm	79.6 cm
Obstacle 2 Length	17 cm	17.2 cm
Obstacle 2 Height	17 cm	17.72 cm
Support's Foot Placement (Heel)	-	22 cm
Swing's Foot Final Placement (Centroid)	-	51.5 cm
Total Execution Time (CV Module)	83.9 ms	-
Mult Value (CFFTG)	-	1.94
Step Height Value (CFFTG)	-	32.7 cm
Number of Iterations (CFFTG)	-	15
Total Execution Time (CFFTG)	800 ms	-

Table 3.5 Experimental Data (e)

Value	**Real Measure**	**Estimated Measure**
Obstacle 1 Distance	32.5 cm	32.71 cm
Obstacle 1 Length	10 cm	10.62 cm
Obstacle 1 Height	10 cm	10.22 cm
Support's Foot Placement (Heel)	-	0 cm
Swing's Foot Final Placement (Centroid)	-	62.5 cm
Total Execution Time (CV Module)	40.8 ms	-
Mult Value (CFFTG)	-	1.76
Step Height Value (CFFTG)	-	30 cm
Number of Iterations (CFFTG)	-	14
Total Execution Time (CFFTG)	410 ms	-

Table 3.6 Experimental Data (f)

4 | Discussion

This chapter will discuss the performance of the pipeline with respect to the results displayed in the previous chapter. It can be easily assessed that the CV module reaches a very good precision when it comes to the geometric evaluation of the obstacles, with a mean error of 0.43 cm between real and estimated measurements. This value is very similar to the one reached by VALOR [57] in its experimental results. It is also clear that the ground plane is present and easily distinguishable, since the clouds obtained after ground plane detection and homogeneous transformation (shown in figures 3.7 , 3.6) are aligned with the Z-axis, in contrast with clouds shown in figures 3.5 and 3.4, which were captured beforehand. This is reasonable, since it was assumed in section 2.1 that the exoskeleton will operate in an indoor environment. This doesn't mean that the algorithm would never be able to recognize the ground plane of an outdoor environment, but the variety of terrains that can be encountered outdoor makes the recognition more challenging. Therefore a more comprehensive and complex approach would be needed to find an accurate solution in that case [75].

The mean execution time of the CV module implemented in C++ is 61.5 ms and, in general, as the size and number of obstacles increases, this time increases as well. Indeed, it can be noted that the computational complexity

of the CV module is dominated by the Track Evaluation function, which has a complexity of $\mathcal{O}(n^2)$ since the score of every point on the tracks is evaluated by comparing the minimum distance between the point and all obstacle points. Even if the execution time happens to be small with respect to the double support time encountered in most walking gaits [76], is still possible to optimize the Track Evaluation in many ways so that this time becomes even smaller. One of the simplest ways to tackle this problem could be the use of a simple 2-approximation algorithm that takes one random point and measures the minimum distance between it and all other points [77]; this would reduce the computational complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. On the other hand, this would return a distance which could be double of the minimum distance returned by the exact solution in the worst case. In the context of cluttered environments, this solution may risk to worsen the performance. A better way to optimize such algorithm would be to produce a 2-D projection of the obstacles on the X-Y plane, and then take only contour points. In such way, the algorithm would have a reduction in the number of points which could be huge in some cases, for example "long and thin obstacles" that occupy a small surface on the X-Y plane, without losing any information regarding the distance between obstacles and possible footholds.

The mean execution time of the CFFTG module implemented in MATLAB is 315 ms. With these values at hand, the maximum double support time can also be evaluated by taking the maximum value of $T(CV) + T(CFFTG)$. It is clear that such value is the one occurring in experiment **(e)**, with a total execution time of 884 ms. This means that, in that case, the exoskeleton will take at least 884 ms between the step depicted in the experiment and the next one. This number is almost surely influenced by the fact that the MATLAB language was not designed with the goal of being time-efficient

and by the complexity of the simulated exoskeleton model. Nevertheless, it is worth highlighting that the complexity of the CFFTG module is $\mathcal{O}(n^2)$, as the CV module, and the number of iterations performed is usually small (on average 14 iterations in all the experimental conditions). Thus, I believe that a more time-efficient implementation of the CFFTG module in C++ under ROS will result in execution time of this algorithm compliant with the real-time application.

All experiments except **(c)** showed that the proposed solution is able to surpass an obstacle in one step; experiment **(c)**, on the other hand, showed that the proposed solution is also able to select smaller steps when the obstacle is too far away to be surpassed, which is the correct behaviour for such situations. In general, obstacles which lies on the closer half of the track with respect to the Y-axis will be stepped over, since the CV module sets its standard foothold in the middle of the track. This can be easily seen in experiment **(c)**, since in that case the CV module decides not to step over the obstacle due to its position on the further half of the track. In this case the CV module decides to place the swing foot at the minimum safe distance from the obstacle. In this way, the next step will have the support foot in the optimal position to step over the obstacle.
Experiments **(a)** and **(b)** shows that the CV module is successful in the presence of obstacles on one or both tracks. Experiment **(d)** is the one where the error on obstacle's height was larger, measuring it as 1.12 cm smaller than its real height (5 cm). Nonetheless, the pipeline was able to correctly surpass the obstacle. Experiment **(e)** highlights a conservative feature of the CV module: evaluation of the minimum distance is based on the detected maximum obstacle height, meaning that smaller obstacles will be treated like the biggest

obstacle on the track. That's why, in the aforementioned experiment, the nearest obstacle, which has an height of 10 cm, is treated in the same way as the further obstacle, which has an height 17 cm, in the minimum distance evaluation phase. Although this behaviour doesn't pose collision problems, in the case of cluttered environments it would most likely discard some feasible footholds. An effective solution to this problem is the segmentation of obstacles, so that every obstacle point is related to a specific obstacle and the minimum distance can be effectively computed for that obstacle only. In this way, the conservative behaviour of the current solution can be relaxed. To effectively segment the obstacle a simple solution is Clustering, which relies on the concept of distance between points (in this case Euclidean) to label them. Early tests on obstacle point clouds already show that this could be an effective way to solve the aforementioned problem. Experiment **(f)** shows that, even when starting with legs in symmetrical position with respect to the X-axis , condition that shouldn't happen since at least one hard-coded step is needed to correctly position the COM, the solution is able to surpass the obstacle. Although it is not shown during the experiments, another improvement lies in the planning of the next steps, which is not part of the current solution, that plans only one step at a time. Since humans and animals naturally move through obstacles planning multiple steps to select the best footholds, so it should be for a Lower Limb Exoskeleton that mimics human locomotion. In this sense, the enhanced solution should output multiple footholds related to the next steps, and evaluate the trajectories between them. This solution would bring more autonomy to the Lower Limb Exoskeleton, which is the ultimate goal of this work.

All experiments show that the trajectories have different mid points and,

in this sense, the proposed approach improves the method used by VALOR based on fixed cubic polynomial trajectories. In the case of VALOR, step length and height are the only parameters used to modify the standard gait pattern. Therefore, the trajectory of the foot during the step is almost the same in every condition. Although this method seems adequate enough to traverse obstacles, humans and animals use very different trajectories based on the shape of the obstacle they have to traverse. Additionally, the implicit limitation imposed by VALOR to surpass an obstacle in more than one step (first the foot is placed at minimum distance, then the next step surpasses the obstacle) is removed for most situations, except when the obstacle is too far to surpass it in one step, as in experiment **(c)**. In addition, my approach allows the exoskeleton to surpass a sequence of obstacles on the foot tracks as soon as they are at a minimum distance from each other (distance which depends on the foot length and the *safeDist* parameter). On the contrary, VALOR's approach assumes that there is at most one obstacle in the forward motion, and that this obstacle is visible at least three steps ahead of the current exoskeleton position.

Another improvement lies in the fact that, in contrast with VALOR's approach, obstacles that don't lie in the tracks have little impact on the execution time, as can be seen in experiment **(d)**. In that case, an additional obstacle appears in the right-most part of the cloud, but the execution time is similar with respect to experiment **(a)**, which has a similar number of obstacle points inside the tracks, as the right-most obstacle is disregarded. On the other hand, VALOR extensively evaluates all obstacle points before considering only those which will actively affect the next steps. This seems unnecessary, since, for example, an obstacle which is placed to the side of the exoskeleton has no effect on its forward motion. Since [57] did not comprise

execution times, those quantities couldn't be compared.
The degree of generalization introduced with respect to the exoskeleton model (section 3.1) allows the use of the proposed solution without loss of generality with respect to the kinematic chain of the specific exoskeleton that will be used for the real implementation.

5 | Conclusions

In this thesis, I proposed and implemented a novel software method to allow powered Lower Limb Exoskeletons to autonomously avoid low obstacles, based on Artificial Intelligence and Computer Vision techniques. This approach allows to overcome the main limitation of current exoskeletons, which are generally driven by pre-defined gait trajectories with no capabilities of context awareness. The proposed solution has showed to be effective in the task of obstacle avoidance, since all simulated experiments have been carried out successfully without collisions. It is also worth highlighting that the exoskeleton model and the assumptions that has been considered in this work allows the portability of the proposed solution to the majority of Lower Limb Exoskeletons on the market or in research, as long as the basic requirements described in sections 2.1 and 3.1 are met. Given the promising results of this thesis, in future work the method will be implemented and tested with a real Lower Limb Exoskeleton worn both with healthy subjects and end-users. I believe that introduction of autonomous behaviors into Lower Limb Exoskeletons may significantly expand their use outside laboratory environments and clinical settings, hopefully increasing their diffusion in daily-living settings and thus enhancing the independence of paraplegic people.

Bibliography

[1] "Paralysed man walks using mind-controlled exoskeleton." https://www.theguardian.com/world/2019/oct/04/paralysed-man-walks-using-mind-controlled-exoskeleton, 2022.

[2] J. A. de la Tejera, R. Bustamante-Bello, R. A. Ramirez-Mendoza, and J. Izquierdo-Reyes, "Systematic review of exoskeletons towards a general categorization model proposal," *Applied Sciences*, vol. 11, no. 1, p. 76, 2020.

[3] "Hardiman wikipedia page." https://en.wikipedia.org/wiki/Hardiman.

[4] H. Lee, P. W. Ferguson, and J. Rosen, *Wearable Robotics*, ch. 11.

[5] Y. He, J. Liu, F. Li, W. Cao, and X. Wu, "Design and analysis of a lightweight lower extremity exoskeleton with novel compliant ankle joints," *Technology and Health Care*, no. Preprint, pp. 1–14, 2021.

[6] B. McGowan, "Industrial exoskeletons: What you're not hearing." https://ohsonline.com/articles/2018/10/01/industrial-exoskeletons-what-youre-not-hearing.aspx, 2018.

[7] R. Y. M. Li and D. P. L. Ng, "Wearable robotics, industrial robots and construction workers safety and health," *Advances in Intelligent Systems and Computing*, pp. 31–36, 2017.

[8] A. S. Koopman, I. Kingma, G. S. Faber, M. P. de Looze, and J. H. van Dien, "Effects of a passive exoskeleton on the mechanical loading of the low back in static holding tasks," *Journal of Biomechanics*, vol. 83, pp. 97–103, 2019.

[9] T. Bosch, J. van Eck, K. Knitel, and M. de Looze, "The effects of a passive exoskeleton on muscle activity, discomfort and endurance time in forward bending work," *Applied Ergonomics*, vol. 54, pp. 212–217, 2016.

[10] J. A. de la Tejera, R. Bustamante-Bello, R. A. Ramirez-Mendoza, and J. Izquierdo-Reyes, "Systematic review of exoskeletons towards a general categorization model proposal," *Applied Sciences*, vol. 11, no. 1, p. 76, 2020.

[11] N. Yagin, "Apparatus for facilitating walking." `https://patents.google.com/patent/US440684`. U.S. Patent.

[12] L. C. Kelley, "Pedomotor." `https://patents.google.com/patent/US1308675`. U.S. Patent.

[13] R. G. Baldovino and R. S. Jamisola, "A survey in the different designs and control systems of powered exoskeleton for lower extremities," *Journal of Mechanical Engineering and Biomechanics*, vol. 1, no. 4, pp. 103–115, 2017.

[14] Y. Mori, J. Okada, and K. Takayama, "Development of a standing style transfer system "able" for disabled lower limbs," *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 4, pp. 372–380, 2006.

[15] 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.

[16] M. Talaty, A. Esquenazi, and J. E. Briceno, "Differentiating ability in users of the rewalk tm powered exoskeleton: An analysis of walking kinematics," in *2013 IEEE 13th international conference on rehabilitation robotics (ICORR)*, pp. 1–5, IEEE, 2013.

[17] 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," in *2011 IEEE international conference on rehabilitation robotics*, pp. 1–8, IEEE, 2011.

[18] "Rex bionics website." `https://www.rexbionics.com/`.

[19] K. Suzuki, Y. Kawamura, T. Hayashi, T. Sakurai, Y. Hasegawa, and Y. Sankai, "Intention-based walking support for paraplegia patient," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 2707–2713, IEEE, 2005.

[20] C. L. Dembia, A. Silder, T. K. Uchida, J. L. Hicks, and S. L. Delp, "Simulating ideal assistive devices to reduce the metabolic cost of walking with heavy loads," *PLOS ONE*, vol. 12, no. 7, p. e0180320, 2017.

[21] K. A. Witte, J. Zhang, R. W. Jackson, and S. H. Collins, "Design of two lightweight, high-bandwidth torque-controlled ankle exoskeletons,"

in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1223–1228, 2015.

[22] J. R. Koller, C. David Remy, and D. P. Ferris, "Comparing neural control and mechanically intrinsic control of powered ankle exoskeletons," in *2017 International Conference on Rehabilitation Robotics (ICORR)*, pp. 294–299, 2017.

[23] A. Esquenazi, M. Talaty, and A. Jayaraman, "Powered exoskeletons for walking assistance in persons with central nervous system injuries: A narrative review," *PM&R*, vol. 9, no. 1, pp. 46–62, 2016.

[24] H. Kazerooni, "Exoskeletons for human power augmentation," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3459–3464, 2005.

[25] J. L. Emken, R. Benitez, and D. J. Reinkensmeyer, "Human-robot cooperative movement training: Learning a novel sensory motor transformation during walking with robotic assistance-as-needed," *Journal of NeuroEngineering and Rehabilitation*, vol. 4, no. 1, 2007.

[26] G. Carpino, D. Accoto, N. Tagliamonte, G. Ghilardi, and E. Guglielmelli, "Lower limb wearable robots for physiological gait restoration: state of the art and motivations," *MEDIC Methodology & Education for Clinical Innovation*, vol. 21, pp. 72–80, 12 2013.

[27] S. Goemaere, M. Van Laere, P. De Neve, and J. M. Kaufman, "Bone mineral status in paraplegic patients who do or do not perform standing," *Osteoporosis International*, vol. 4, no. 3, pp. 138–143, 1994.

[28] T. Sumiya, K. Kawamura, A. Tokuhiro, H. Takechi, and H. Ogata, "A survey of wheelchair use by paraplegic individuals in japan. part 2:

Prevalence of pressure sores," *Spinal Cord*, vol. 35, no. 9, pp. 595–598, 1997.

[29] K. A. Strausser, T. A. Swift, A. B. Zoss, H. Kazerooni, and B. C. Bennett, "Mobile exoskeleton for spinal cord injury: Development and testing," in *Dynamic Systems and Control Conference*, vol. 54761, pp. 419–425, 2011.

[30] "Ekso website." `https://eksobionics.com/`.

[31] W. Y.-W. Tung, M. McKinley, M. V. Pillai, J. Reid, and H. Kazerooni, "Design of a minimally actuated medical exoskeleton with mechanical swing-phase gait generation and sit-stand assistance," in *Dynamic Systems and Control Conference*, vol. 56130, p. V002T28A004, American Society of Mechanical Engineers, 2013.

[32] E. WS, "Center of mass of the human body helps in analysis of balance and movement," *MOJ Applied Bionics and Biomechanics*, vol. 2, no. 2, 2018.

[33] J.-Y. Jung, W. Heo, H. Yang, and H. Park, "A neural network-based gait phase classification method using sensors equipped on lower limb exoskeleton robots," *Sensors*, vol. 15, no. 11, pp. 27738–27759, 2015.

[34] H. A. Quintero, R. J. Farris, and M. Goldfarb, "Control and implementation of a powered lower limb orthosis to aid walking in paraplegic individuals," in *2011 IEEE International Conference on Rehabilitation Robotics*, pp. 1–6, IEEE, 2011.

[35] R. J. Farris, H. A. Quintero, and M. Goldfarb, "Performance evaluation of a lower limb exoskeleton for stair ascent and descent with paraple-

gia," in *2012 Annual international conference of the IEEE engineering in medicine and biology society*, pp. 1908–1911, IEEE, 2012.

[36] T. Bacek, M. Moltedo, K. Langlois, G. A. Prieto, M. C. Sanchez-Villamañan, J. Gonzalez-Vargas, B. Vanderborght, D. Lefeber, and J. C. Moreno, "Biomot exoskeleton—towards a smart wearable robot for symbiotic human-robot interaction," in *2017 International Conference on Rehabilitation Robotics (ICORR)*, pp. 1666–1671, IEEE, 2017.

[37] D. Winter, "Human balance and posture control during standing and walking," *Gait & Posture*, vol. 3, no. 4, pp. 193–214, 1995.

[38] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005.

[39] T. McGeer, "Passive dynamic walking," *The International Journal of Robotics Research*, vol. 9, no. 2, pp. 62–82, 1990.

[40] T. Kagawa and Y. Uno, "Gait pattern generation for a power-assist device of paraplegic gait," in *RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 633–638, IEEE, 2009.

[41] T. Kagawa and Y. Uno, "A human interface for stride control on a wearable robot," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4067–4072, IEEE, 2009.

[42] S. Wang, L. Wang, C. Meijneke, E. Van Asseldonk, T. Hoellinger, G. Cheron, Y. Ivanenko, V. La Scaleia, F. Sylos-Labini, M. Molinari,

et al., "Design and control of the mindwalker exoskeleton," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 23, no. 2, pp. 277–286, 2014.

[43] L. Wang, S. Wang, E. H. van Asseldonk, and H. van der Kooij, "Actively controlled lateral gait assistance in a lower limb exoskeleton," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 965–970, IEEE, 2013.

[44] R. Kolaghassi, M. K. Al-Hares, and K. Sirlantzis, "Systematic review of intelligent algorithms in gait analysis and prediction for lower limb robotic systems," *IEEE Access*, 2021.

[45] Y. Sankai, "Hal: Hybrid assistive limb based on cybernics," in *Robotics research*, pp. 25–34, Springer, 2010.

[46] A. Tsukahara, Y. Hasegawa, K. Eguchi, and Y. Sankai, "Restoration of gait for spinal cord injury patients using hal with intention estimator for preferable swing speed," *IEEE Transactions on neural systems and rehabilitation engineering*, vol. 23, no. 2, pp. 308–318, 2014.

[47] S. Wang, W. Van Dijk, and H. van der Kooij, "Spring uses in exoskeleton actuation design," in *2011 IEEE International Conference on Rehabilitation Robotics*, pp. 1–6, IEEE, 2011.

[48] F. Sylos-Labini, V. La Scaleia, A. d'Avella, I. Pisotta, F. Tamburella, G. Scivoletto, M. Molinari, S. Wang, L. Wang, E. van Asseldonk, *et al.*, "Emg patterns during assisted walking in the exoskeleton," *Frontiers in human neuroscience*, vol. 8, p. 423, 2014.

[49] K. Kong and D. Jeon, "Design and control of an exoskeleton for the elderly and patients," *IEEE/ASME Transactions on mechatronics*, vol. 11, no. 4, pp. 428–432, 2006.

[50] Z. Li, Y. Yuan, L. Luo, W. Su, K. Zhao, C. Xu, J. Huang, and M. Pi, "Hybrid brain/muscle signals powered wearable walking exoskeleton enhancing motor ability in climbing stairs activity," *IEEE Transactions on Medical Robotics and Bionics*, vol. 1, no. 4, pp. 218–227, 2019.

[51] S. Guo, Y. Ding, and J. Guo, "Control of a lower limb exoskeleton robot by upper limb semg signal," in *2021 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 1113–1118, IEEE, 2021.

[52] S. Kyeong, J. Feng, J. K. Ryu, J. J. Park, K. H. Lee, and J. Kim, "Surface electromyography characteristics for motion intention recognition and implementation issues in lower-limb exoskeletons," *International Journal of Control, Automation and Systems*, vol. 20, no. 3, pp. 1018–1028, 2022.

[53] X. Zhao, W.-H. Chen, B. Li, X. Wu, and J. Wang, "An adaptive stair-ascending gait generation approach based on depth camera for lower limb exoskeleton," *Review of Scientific Instruments*, vol. 90, no. 12, p. 125112, 2019.

[54] F. Xu, R. Huang, H. Cheng, J. Qiu, S. Xiang, C. Shi, and W. Ma, "Stair-ascent strategies and performance evaluation for a lower limb exoskeleton," *International Journal of Intelligent Robotics and Applications*, vol. 4, no. 3, pp. 278–293, 2020.

[55] B. Laschowski, W. McNally, A. Wong, and J. McPhee, "Preliminary design of an environment recognition system for controlling robotic lower-

limb prostheses and exoskeletons," in *2019 IEEE 16th international conference on rehabilitation robotics (ICORR)*, pp. 868–873, IEEE, 2019.

[56] K. Struebig, N. Ganter, L. Freiberg, and T. C. Lueth, "Stair and ramp recognition for powered lower limb exoskeletons," in *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1270–1276, IEEE, 2021.

[57] D.-X. Liu, J. Xu, C. Chen, X. Long, D. Tao, and X. Wu, "Vision-assisted autonomous lower-limb exoskeleton robot," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, pp. 3759–3770, 2021.

[58] "Decree of june 14th 1989: Italian version." `https://www.bosettiegatti.eu/info/norme/statali/1989_0236.htm#08.1.10`.

[59] K. G. Derpanis, "Overview of the ransac algorithm," *Image Rochester NY*, vol. 4, no. 1, pp. 2–3, 2010.

[60] T. Strutz, *Data fitting and uncertainty*. 2016.

[61] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE international conference on robotics and automation*, pp. 1–4, IEEE, 2011.

[62] S. M. LaValle, "Homogeneous transformation matrices." `http://planning.cs.uiuc.edu/node111.html`. Cambridge University.

[63] L.-S. Chou and L. F. Draganich, "Placing the trailing foot closer to an obstacle reduces flexion of the hip, knee, and ankle to increase the risk of tripping," *Journal of biomechanics*, vol. 31, no. 8, pp. 685–691, 1998.

[64] L.-S. Chou, K. R. Kaufman, R. H. Brey, and L. F. Draganich, "Motion of the whole body's center of mass when stepping over obstacles of different heights," *Gait & Posture*, vol. 13, no. 1, pp. 17–26, 2001.

[65] A. Tsoularis and C. Kambhampati, "On-line planning for collision avoidance on the nominal path," *Journal of Intelligent and Robotic Systems*, vol. 21, pp. 327–371, 04 1998.

[66] S. Sharif Bidabadi, I. Murray, and G. Y. F. Lee, "Validation of foot pitch angle estimation using inertial measurement unit against marker-based optical 3d motion capture system," *Biomedical Engineering Letters*, vol. 8, no. 3, pp. 283–290, 2018.

[67] H. H. Hoos and E. Tsang, "Chapter 5 - local search methods," in *Handbook of Constraint Programming* (F. Rossi, P. van Beek, and T. Walsh, eds.), vol. 2 of *Foundations of Artificial Intelligence*, pp. 135–167, Elsevier, 2006.

[68] J. Carpentier, M. Benallegue, and J.-P. Laumond, "On the centre of mass motion in human walking," *International Journal of Automation and Computing*, vol. 14, no. 5, pp. 542–551, 2017.

[69] "Crank connecting rod equations." `https://www.istitutopesenti.edu.it/dipartimenti/meccanica/meccanica/biella.pdf`.

[70] "Intel realsense d455 web page." `https://www.intelrealsense.com/depth-camera-d455/`.

[71] B. Siciliano and O. Khatib, *Springer Headbook of Robotics*, ch. 22.

[72] "Intel realsense beginners guide to depth." `https://www.intelrealsense.com/beginners-guide-to-depth/`.

[73] "What is ros?." https://ubuntu.com/robotics/what-is-ros.

[74] "Ros wiki." https://wiki.ros.org/ROS/Tutorials.

[75] A. A. Rafique, A. Jalal, and K. Kim, "Statistical multi-objects segmentation for indoor/outdoor scene detection and classification via depth images," in *2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pp. 271–276, IEEE, 2020.

[76] A. Kharb, V. Saini, Y. Jain, and S. Dhiman, "A review of gait cycle and its parameters," *IJCEM International Journal of Computational Engineering & Management*, vol. 13, pp. 78–83, 2011.

[77] M. Dalirrooyfard, V. V. Williams, N. Vyas, N. Wein, Y. Xu, and Y. Yu, "Approximation algorithms for min-distance problems," 2019.

[78] L.-F. Zhang, Y. Ma, C. Wang, Z. Yan, and X. Wu, "A method for arm motions classification and a lower-limb exoskeleton control based on semg signals," in *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*, pp. 118–123, 2019.

[79] A. A. Bakri, M. Y. Lezzar, M. Alzinati, K. Mortazavi, W. Shehieb, and T. Sharif, "Intelligent exoskeleton for patients with paralysis," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 189–193, 2018.

[80] I. Tijjani, S. Kumar, and M. Boukheddimi, "A survey on design and control of lower extremity exoskeleton for bipedal walking," *Applied Sciences*.

[81] "Intel realsense d455 camera sdk." https://github.com/IntelRealSense/librealsense/releases/tag/v2.50.0.

[82] "Intel realsense ros package." https://github.com/IntelRealSense/realsense-ros.

[83] K. Junius, B. Brackx, V. Grosu, H. Cuypers, J. Geeroms, M. Moltedo, B. Vanderborght, and D. Lefeber, "Mechatronic design of a sit-to-stance exoskeleton," in *5th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, pp. 945–950, 2014.

[84] M. K. Shepherd and E. J. Rouse, "Design and validation of a torque-controllable knee exoskeleton for sit-to-stand assistance," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 4, pp. 1695–1704, 2017.

[85] "What is matlab?." https://it.mathworks.com/discovery/what-is-matlab.html.

[86] "Matlab walking robot repository." https://github.com/mathworks-robotics/msra-walking-robot.

Acknowledgements

I'd like to thank my family and my friends for sticking with me during this journey, during the highs and the lows. I also thank Prof. Menegatti, Dr. Tortora and the University of Padua for giving me the opportunity to expand my knowledge and giving me the tools to produce this thesis. Without them, all of this couldn't have been possible.