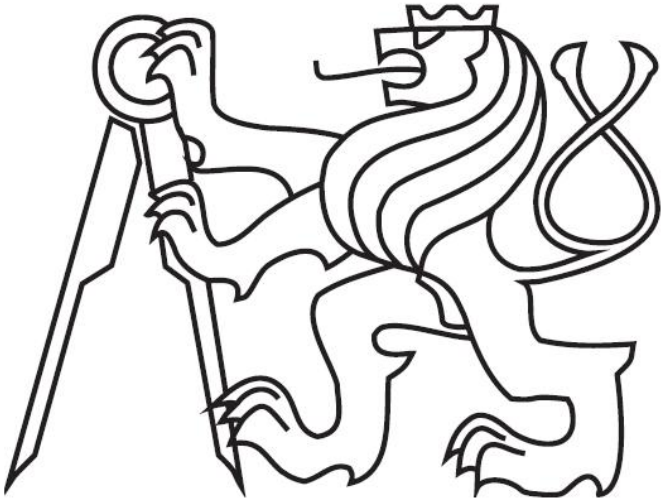
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF ELECTRICAL ENGINEERING
Department of Cybernetics

BACHELOR THESIS



Diar Masri

Motion Model of Stair Climbing in Hexapod Walking Robot

Supervisor: Ing. Jan Faigl, Ph.D.
Prague, 2014

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne:..... Diar Masri

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Diar M a s r i

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Model pohybu kráčejícího šestinohého robotu při chůzi po schodech

Pokyny pro vypracování:

1. Seznamte se s robotickou platformou kráčejícího robotu a způsoby jeho řízení.
2. Vytvořte model a rozhraní pro simulaci robotické platformy v prostředí V-REP.
3. Seznamte se s principem RGB-D kamer a prostřednictvím zařízení Asus Xtion PRO a knihovny PCL (Point Cloud Library) vytvořte parametrický a geometrický model schodů.
4. Navrhněte model pohybu robotu kráčejícího po schodech nahoru.
5. Ověřte navržený model simulací a reálným experimentem.
6. Uvažujte omezení navrženého modelu pro chůzi ze schodů a diskutujte možná rozšíření s využitím dalších senzorických vstupů.

Seznam odborné literatury:

[1] Point Cloud Library (PCL, http://pointclouds.org/
[2] Shibendu Shekhar Roy, Dilip Kumar Pratihar: Modeling and analysis of six-legged robots: Analytical and soft computing-based methods. Lambert Academic Publishing, 2012.
[3] E. Z. Moore and M. Buehler: Stable Stair Climbing in a Simple Hexapod Robot. McGill University, 2001.
[4] Steven M. Lavalle: Planning Algorithms. Cambridge University Press, May 2006.
[5] Ya-Cheng Chou, Wei-Shun Yu, Ke-Jung Huang and Pei-Chun Lin: Bio-inspired step-climbing in a hexapod robot. Bioinspir. Biomim. 7 (2012) 036008.

Vedoucí bakalářské práce: Ing. Jan Faigl, Ph.D.

Platnost zadání: do konce zimního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 25. 10. 2013

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Diar M a s r i

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Bachelor Project: Motion Model of Stair Climbing in Hexapod Walking Robot

Guidelines:

1. Familiarize yourself with the hexapod walking robot and its control.
2. Develop a model and programming interface for simulation of the platform in the V-REP framework.
3. Study principle of RGB-D cameras and use Asus Xtion PRO together with the PCL (Point Cloud Library) to create a parametrized model and geometrical model of the stairs.
4. Propose a model of motion control for climbing up the stairs.
5. Verify the proposed model in simulations and real experiment.
6. Consider limitations of the designed model for climbing down the stairs and discuss its possible extension using additional sensoric information.

Bibliography/Sources:

[1] Point Cloud Library (PCL, http://pointclouds.org/
[2] Shibendu Shekhar Roy, Dilip Kumar Pratihar: Modeling and analysis of six-legged robots: Analytical and soft computing-based methods. Lambert Academic Publishing, 2012.
[3] E. Z. Moore and M. Buehler: Stable Stair Climbing in a Simple Hexapod Robot. McGill University, 2001.
[4] Steven M. Lavalle: Planning Algorithms. Cambridge University Press, May 2006.
[5] Ya-Cheng Chou, Wei-Shun Yu, Ke-Jung Huang and Pei-Chun Lin: Bio-inspired step-climbing in a hexapod robot. Bioinspir. Biomim. 7 (2012) 036008.

Bachelor Project Supervisor: Ing. Jan Faigl, Ph.D.

Valid until: the end of the winter semester of academic year 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, October 25, 2013

Abstrakt

Cílem této práce je rozšíření pohybových možností šestinohé kráčející robotické platformy o pochyb po schodech. Dobrá prostupnost složitým terénem patří mezi hlavní výhody kráčejících robotů. To je dáno především vyšším počtem stupňů volnosti než třeba kolové roboty typu auta, na druhou stranu je plánování pohybu těchto robotů v plné obecnosti velmi složitou a náročnou úlohou. Proto se při návrhu modelu pohybu po schodech vycházelo ze způsobu chůze navrženého pro pohyb robotu po rovinně. Takto navržený model pohybu byl následně parametrizován pro různé rozměry schodů, která má robot vylézt.

Dále je součástí bakalářské práce odhad parametrů schodů, které se nacházejí před robotem a které snímá hloubkovou kamerou. Po automatické identifikaci rozměrů schodů je zvolen vhodný způsob pro jejich překonání již bez zpětné vazby z prostředí. Navržený model pohybu po schodech nahoru byl implementován a experimentálně ověřen pro robotickou platformu PhantomX. V experimentech byl robot schopen zdolat schody různých rozměrů, ale také se ukázalo, že v případě automatického rozpoznávání schodů z jediného snímku schodů, závisel úspěch pohybu na správnosti odhadu jejich rozměrů.

Abstract

The goal of this thesis is to extend the motion capabilities of the hexapod walking robot that will allow it to climb stairs located in front of the robot. A high traversability of this type of robots in rough terrain is one of the main feature of walking robots. However, it is at the cost of a higher complexity of their motion control because of a high number of degrees of freedom. Therefore, the developed motion gait is based on a regular gait for motion on planar surfaces. The proposed motion model has been further parametrized for various stairs, which the robot should traverse.

Then, an automatic identification of the stairs parameters have been studied using a single image of the stairs taken by a RGB-D camera. After an estimation of the parameters, the most suitable stair climbing gait is automatically selected and the robot traverses the stairs. The proposed motion gaits and image processing have been implemented and experimentally verified with the PhantomX hexapod walking robot. The robot has been able to traverse different stairs and provides experimental evidence of the proposed gaits. During the experiments, we also found out that the success of the mission depends on estimated parameters and because the parameters are estimated from a single image, it is necessary to provide a clear view on the stairs.

Acknowledgments

I would like to thank my thesis supervisor, Ing. Jan Faigl, Ph.D., for his patience and advice while working on this thesis. And also to Ing. Petr Vaněk for help regarding the work on the robot.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

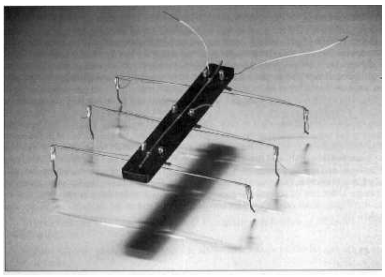
A wide range of unmanned ground vehicles have been developed to address specific requirements, such as speed and traversability of different terrain. Regarding traversability capabilities in complex and rough terrains, one of the most attractive platforms are walking robots. Their most valuable feature considering the topic of traversing unstructured environment is their capabilities to deal with unstructured environment such as rough terrain, debris, etc.

The legged platforms are of various types . A design and also control of legged robotic platforms are inspired by nature. There is rapidly developing research in two-legged platforms that are inspired by humanoids [2], there is also research focused on quadruped robots [3] and their locomotion over rough terrain. The quadruped robots are inspired by four-legged mammals. In this thesis,we are focused on six-legged robots, also called hexapods. These platforms differ from quadruped robots in the sense of providing more stability due to a higher number of legs. The six-legged robots are sometimes miscalled spiders, but in nature spider has eight legs and they are represented in robotics as octopods. Hexapods are inspired by ants that use six legs for their locomotion.

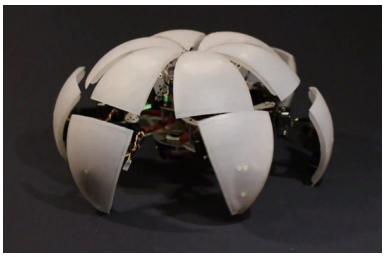
There are various six-legged platforms that differ from each other. The difference between them is due to physical constitution of their body, some of them have rectangular body, which makes it able to recognize direction of its movement, e.g., the platform Stiquito showevd in Figure 1.0.1a, and some of them are circular, platform MorpHex in Figure 1.0.1b, which has the same proportions from each point of view. The difference can be also made in movement possibilities of the leg, as there are robots with legs that are very similar to insect world like the platform PhantomX in Figure 1.0.1c, or robots that use rotation of leg around its axis, the platform RHex [4] in Figure 1.0.1d.

The motion of these crawlers is controlled by different approaches. In general, there are two approaches that can be identified in literature to control a motion of hexapod robots. The first is inspired by a biological neural network that is utilized as the so called central pattern generator (CPG). The second approach is inspired by a pattern of movement and its transfer into repeated moves called gaits. The goal of this thesis is to develop an advanced gait that will make the robot able to traverse stairs.

Several gaits have been proposed in [6] for a robot motion on a planar surface, which

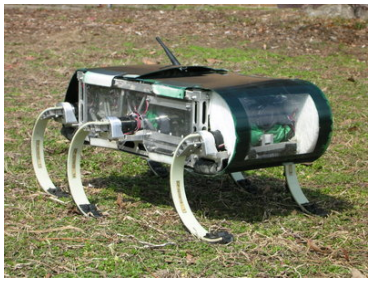


(a) Stiquito



(b) MorpHex [5]

(c) PhantomX Mark II



(d) RHex [4]

Figure 1.0.1: Hexapod platforms

are able to move the platform in desired directions. They differ in an order in which the legs move and in the number of legs that the robot is standing on while moving. The reason for developing an advanced gait is that the algorithm for motion on the planar surface does not work when approaching an obstacle. The gait developed for motion on a planar surface is designed to be fast and power efficient, and thus, it is not designed to move legs to a height that is higher than the needed for a forward motion, because it takes more time and energy. So, the leg moves in the least possible height, for safe and fast forward motion. Hence, when the robot is approaching an obstacle, the legs are moving low when transferring forward and thus hit the obstacle instead of landing on it.

This could be solved by transferring each leg to a higher level, but when the robot is on stairs, there could easily be done a mistake in landing the foot, which would lead to slip and change in the direction of the movement and a possible fall.

However, in this naïve approach, the robot does not have a full information about its presence on the stairs. The robot only knows that its feet are supposed to move higher and it does not land the foot in appropriate position. The lack of the information results in a mistake in landing the foot, and therefore, a different gait has to be designed to address this issue.

It is clear that physical constitution of the robot does not allow to traverse any stairs, but it can be expected that a relatively wide range of stairs with different dimensions can be traverse in the same manner like humans are able to traverse various stairs.

Chapter 2

Problem statement

The problem addressed in this thesis is motivated by enhancing motion capabilities of six-legged walking robots by a new motion control gait for traversing stairs. The stairs are supposed to be a rigid well defined object that can be parametrized by dimensions of identical stairs. We aim to develop a parametrized gait for crawling stairs and experimentally verify these new crawling capabilities in practical experiments. The motion gait for stairs crawling is supposed to provide a sequence of motions for selected parameters of stairs. This will make the motion gait general enough to traverse different stairs that the robot is capable to traverse.

The parameters of the stairs maybe given. However, we rather leverage on current advancements on RGB-D camera and propose estimation of the parameters based on depth image provided by a commercial RGB-D Asus Xtion Pro [7]. Therefore, we also aim to develop an algorithm for depth image processing providing the required parameters from a real image of the stairs.

Motion control of the robot is a sequence of specific motion actions. In this thesis, the robot is a walking robotic platform with six legs and each leg has three degrees of freedom (DoF). This feature makes the robot able to put each of its feet in variety of positions. Which however, makes the robot control a difficult task, as the number of DoF is 18. The robot itself can be seen as an object in space, which turns out to additional coordinates describing the robot position in 3D word and three angular coordinates describing the robot yaw, pitch, and roll. So, it makes the full system with 24 degrees of freedom in total.

In this thesis, the problem is to develop a motion control algorithm that will allow the robot to traverse an obstacle in a form of stairs. The stairs are considered to be defined by three parameters (*h,d,w*), see Figure 2.1.1. These parameters are the same for each step in the stairs. There is also a parameter N, which represents the number of the stairs to be traversed. There should be no limits for the parameter N, despite the fact that stairs are formed from at least two particular stairs, as we are referring to stairs, which are usually formed of more than one stair. By using more than one stair, motion capabilities of the platform enabled by the proposed algorithm will be fully demonstrated.

2.1 Stairs

Stairs are supposed to be defined by four parameters:

h – height [mm],

d – depth [mm],

w – width [mm],

N – number of stairs [-],

which are depicted in Figure 2.1.1.

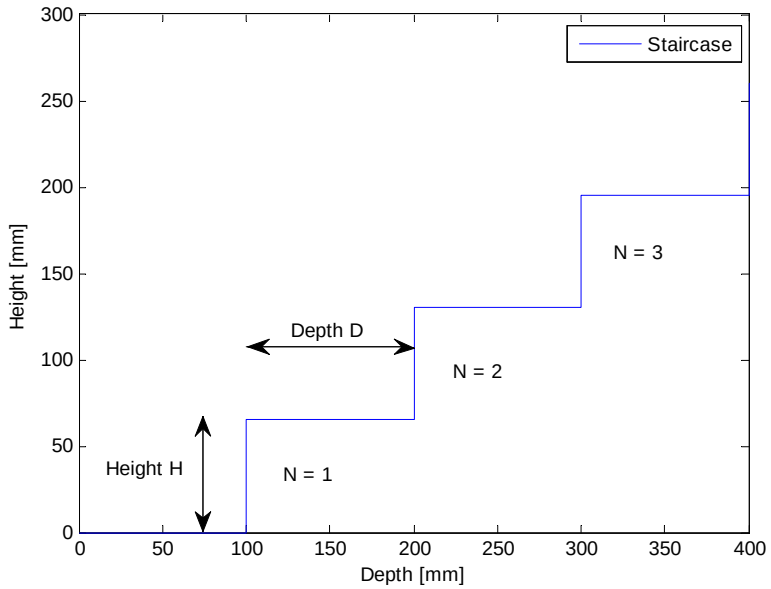


Figure 2.1.1: Parameters D and H of stairs

It is assumed that each parameter (h,d,w) is identical for each step of the stairs. The most important parameters for the motion control are h and d. Another condition that has to be fulfilled it that the stairs have to be wide enough for the robot to traverse it. Thus, the parameter w has to satisfy condition Equation 2.1.1.

$$w > w_{min}, \tag{2.1.1}$$

Where w_{min} is a minimum width of the stairs that allows the robot to safely traverse them. The minimum value of the stairs relates to the width of the robot and in the case of the platform used in this thesis, it is:

$$w_{min} = 600.$$

It is supposed that the last stair is followed by a platform, large enough to provide a support for the robot to stand on all feet, which will mark the end of the stair climbing.

Chapter 3

Hardware

The robot used for the purpose of verifying the functionality of the proposed solution is the robotic platform PhantomX Mark II [8], which is a six legged walking robot. Each joint of the robot is controlled by Dynamixel actuator, which is an intelligent actuator driver connected to a serial bus and capable of performing different control commands. A detailed description of the drive is presented in the next section.

3.1 Dynamixel AX-18A actuator

The Dynamixel actuators [9] are intelligent actuators that are made to control the motion of the robot. They do not work only as simple actuators, but they are also able to work as intelligent sensors. The actuator is controlled by microcontroller unit (MCU) and communicates through 3-wire connection, which is also used to power supply it.

(a) Body of the actuator

(b) Operation range of actuator

Figure 3.1.1: Dynamixel AX-18A actuator

3.1.1 Hardware

A body of the actuator is made of engineering plastic, which is very resistant material. On the front side (see Figure 3.1.1a), there are visible two ports for a connection; so, the actuator can be easily connected to a serial bus. The physical features of the Dynamixel actuator are listed in Table 3.1.

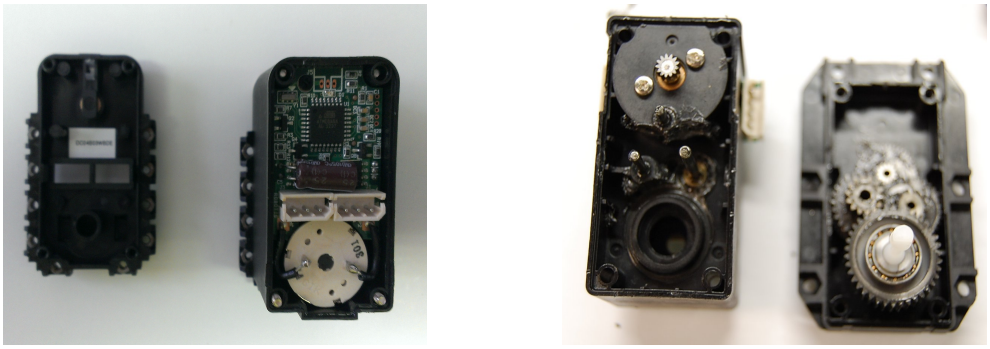
Table 3.1: Parameters of the Dynamixel AX-18A actuator

Model	**AX-18A**
Stall Torque	1.8 N/m
Speed (RPM)	97
Nominal Operating Value	12 V
Stall Current Draw	2.2 A
Dimensions	32x50x40 mm
Weight	54.5 g
Resolution	0.29º
Operating Angle	300
Gear Reduction	254:1
CPU	ATMega8
Position Sensor	Potentiometer
Com Protocol	TTL
Com Speed	1 MBps

A servomotor is inside the actuator, which propels the system of gears. They are all made of plastic; however in the case of AX-18A model, the first sprocket is made of steel, which is supposed to make the movement of the actuator faster. The rest of the actuators is still made of plastic; so, when a higher torque is applied on them, the plastic sprockets are not able to deal with the amount of torque and are crushed. There is also the microcontroller unit (MCU) ATMega 8. The whole inside is shown in Figure 3.1.2a and 3.1.2b.

Despite small dimensions of the actuator, it is capable of a high torque and high speed, at the cost of a high power consumption, see specification list in Table 3.1. The current consumption of a single actuator in a standby mode should not be higher than 50 mA (according to the manufacturer specifications); however, during the movement of the robot, the value can vary between 50 and 2200 mA. This value mainly depends on the difference between the current position and the desired position, but also on the present load on the actuator.

This is the reason, why the operational time of the robot is short when it is powered from a battery pack. The main problem is that the robot consumes energy not only when moving, but also when it does not move, because the actuators have to stay at the desired position. The total power consumption can be even higher in the case when all the actuators are



(a) Front Side (b) Rear Side

Figure 3.1.2: Inside of the Dynamixel AX-18A actuator

operating simultaneously. Because of the needed power supply a design of energy efficient motion control is being further studied [10].

Regarding the power supply and communication, each actuator is connected via three wires. One wire is used for power supply and the two others are used for communication based on the transistor–transistor logic (TTL).

The actuators are able to set their goal position in range between -150 and 150 degrees around the zero position as it can be seen in Figure 3.1.1b. Settings of the goal position is precise as the resolution is nearly third of the degree.

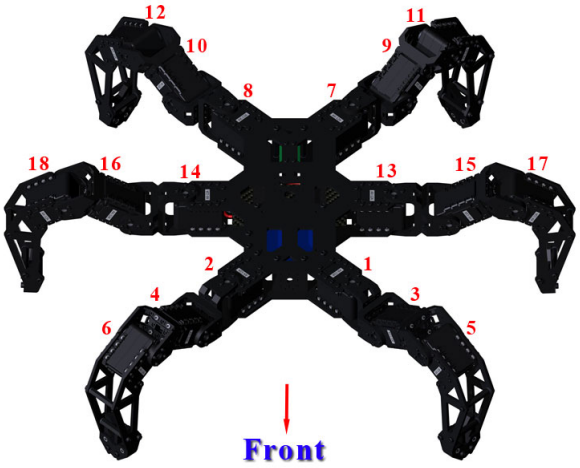


Figure 3.1.3: Positions of the platform's actuators

3.1.2 Communication

These actuators represent intelligent actuators controlled by the TTL and the control unit must support the half duplex universal asynchronous receiver and transmitter (UART) serial

communication. The actuators are connected together by 3-wire connection. The actuators are connected through serial communication and each actuator is being addressed by its unique ID. The ID of each actuator can be modified; however, the motion control commands refer to actuators according to the schema shown in Figure 3.1.3.

The communication between control unit and actuator is realized by two types of packets: 1) instruction packet; 2) status packet. Each packet is divided into two parts: the header part and the data part.

Figure 3.1.4: Instruction packet

Instruction packet definition can be seen in Figure 3.1.4, where each box represents one byte. The header is formed by the first two bytes, which are used to mark the beginning of the packet. The data part consists of ID of the addressed actuator, the length of the whole packet, the instruction for the addressed actuator, parameters requested for particular instruction and a checksum. The instruction defines whether the parameters are for read, write or synchronous write to more actuators at one time. Parameters are optional data, for example an address of the data to be read. The list of most used addresses is shown in Table 3.2.

Table 3.2: Common addresses of the actuators

Address	**Item**	**Length (bytes)**	**Min**	**Max**
3	ID	1	0	253
4	Baud Rate	1	0	254
5	Return Delay Time	1	0	254
6	CW Angle Limit	2	0	1023
8	CCW Angle Limit	2	0	1023
14	Max Torque	2	0	1023
17	Alarm LED	1	0	127
18	Alarm Shutdown	1	0	127
30	Goal Position	2	0	1023
32	Moving Speed	2	0	1023

The checksum is computed as an inverted sum of the bytes in the data part. For the synchronous write, the ID is set to a broadcasting level and there is not checksum as the communication is through broadcasted.

The status packet is shown in Figure 3.1.5, where each box represents one byte. The structure of the status packet is similar to the structure of the instruction packet. The header

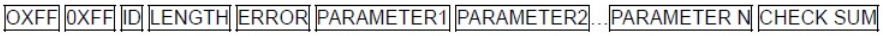


Figure 3.1.5: Return packet

is formed by the first two bytes, which are used to mark the beginning of the packet. The data part consists of ID of previously addressed actuator, length of the whole packet, possible error in actuator, parameters requested for particular instruction and checksum. In the case of no problem in communication, nothing is being transferred in the error byte. If an error appears, there is an information about it and its origin. The MCU in the actuator can differ between error that happened because of, e.g., overheat of the actuator, not enough input voltage, etc. All the return values are described in Table 3.3.

Table 3.3: Error details

Bit	Name	Detail
7	0	-
6	Instruction Error	Set to 1 if an undefined instruction is sent.
5	Overload Error	Set to 1 if the specified maximum torque can't control the applied load.
4	Checksum Error	Set to 1 if the checksum of the instruction packet is incorrect.
3	Range Error	Set to 1 if the instruction sent is out of the defined range.
2	Overheat Error	Set to 1 if the internal temperature of the Dynamixel unit is above the operating temperature range as defined in the control table.
1	Angle Limit Error	Set as 1 if the Goal Position is set outside of the range between CW Angle Limit and CCW Angle Limit.
0	Input Voltage Error	Set to 1 if the voltage is out of the operating voltage range as defined in the control table.

Maximum number of actuators in a single serial bus is 254, as the possible ID is in range 0-253, the ID number 254 is used for broadcast communication when information is addressed to all connected actuators at one moment. This limitation arise from the fact, that ID is represented as a byte, and number 255 is reserved for start byte.

There are different baud rates that can be set for communication. The computation of baud rate is done by Equation 3.1.1

$$Speed = 2000000/(Address4 + 1), \tag{3.1.1}$$

which sets the baud rate value, where the maximum baud rate error of 3% is within tolerance of the UART communication. The initial value is set to 1.

The Dynamixel actuator does not work only as a simple actuator, but is can also act as an intelligent sensor. When any of the parameters exceeds the limit, which is set for the particular actuator, the actuator is disconnected from the system. If the reason is temporary, such as exceeding the angle limit, the actuator can move again, when the next goal position, which is inside the limits, is received. There are however problems that can be solved only by restarting the actuator, for example exceeding the maximum load. Simultaneously to a report a LED diode lights to inform, e.g., a collision occurred on the actuator.

The communication between control unit and the robot can be done in several ways.

- Microcontroller unit - used in the original kit, where control was done by remote controller [11] that gave instructions for Arbotix board [12] (MCU). The actuators are connected directly to the MCU.
- Xbee communication - a wireless communication,where control unit sends data to Xbee transmitter and Xbee receiver is on the robot. This does not directly support half-duplex communication.
- USB - A control unit sends a command through USB cable, which is transferred to the TTL.

In this thesis, the last way of the communication is used for a reliable communication. The only problem can be a limited length of cables. The conversion between USB and TTL is done by the USB2DYNAMIXEL adapter. During the testing of the whole process of the stair climbing, the Xbee communication was used, because there is no problem with length of the cables.

3.2 Robotic platform PhantomX Mark II

Figure 3.2.1: Robotic platform PhantomX Mark II

The robotic platform PhantomX Mark II [8] is a six legged walking robot with dimensions of the body 240×120×39 mm, the robot is shown in Figure 3.2.1. Parts of the robot are made of rugged plexiglass, which makes it more resistant to hits and falls. Each leg has

three degrees of freedom, which allows it to place the foot in various positions. However, there are limits for the movement, which arise from the construction of the leg.

Each leg consists of three Dynamixel AX-18A actuators. Legs on one side are constructed in a same way and legs on the opposite side are constructed mirrored. Each actuator of a single leg has associated name to easily manipulate with each leg and address the actuators, as it can be seen in Figure 3.2.2. The name for each actuator is from Latin name of the particular parts of an insect body, which has similar motion as the.

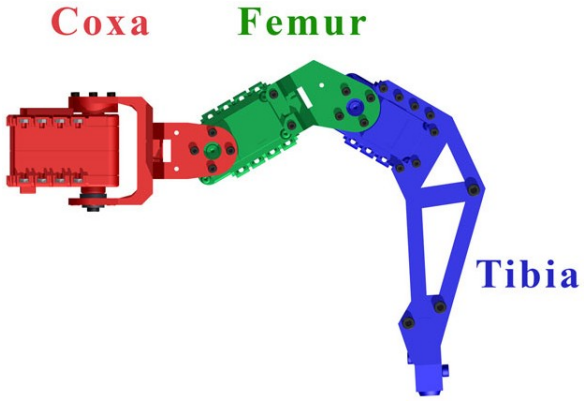


Figure 3.2.2: Names of each part of leg

3.2.1 Practical issues

We made following modification of the original PhantomX platform to make it usable for the experimenting with the proposed gaits for crawling upstairs. The original version was controled by Arbotix board [12] and remote controller [11] that gave instructions for Arbotix board (MCU). The actuators are connected directly to MCU, which has no usage in this task, as the main feature should be autonomous.

As the power supply was used LiPol battery with 2200 mAh, but it could supply the robot for about fifteen minutes. Due to the amount of time spent on charging the battery, which was about two hours, it seems very unpractical to use this solution for testing and development of gaits. That is why we added a possibility to power the robot from a power supply, which makes the testing faster and more comfortable.

However, there are still couple of issues to be aware of, like length of the supply cables or a high current which can be dangerous; so all cables should be properly isolated. Another limitation comes from the Dynamixel actuators, as they are powered even if the robot does not move and thus they have tendency to overheat in time.

3.3 ASUS Xtion PRO camera

The ASUS Xtion PRO [7] is a RGB-D camera considered for automatic estimation of parameters of the stairs the robot is requested to crawl. It is a system that provides multiple

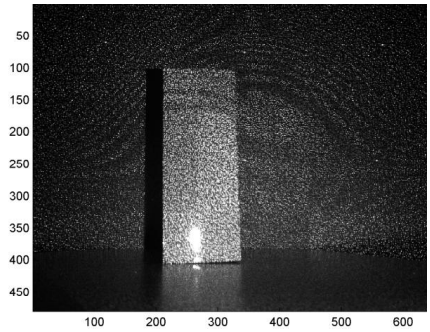
Figure 3.3.1: ASUS Xtion PRO

sensing functions. It was originally designed for motion-sensing application and games; thus, the operation environment is indoor. It is able to provide RGB images, depth images and also includes microphone to detect sound. The camera is connected via USB 2.0 interface, which also provides power. The depth images provided by the camera are very useful considering the addressed task, because it is possible to calculate the parameters of the stairs (h,d,w,N) from the depth image.

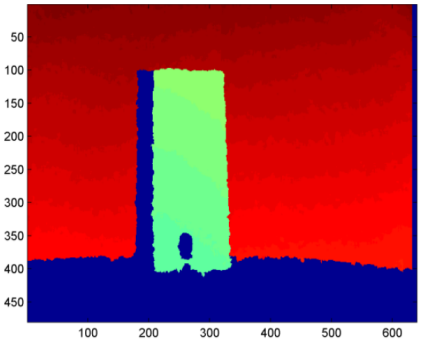
The depth sensing is based on infrared (IR) projector and IR sensor. The camera consists of one color image sensor and one IR sensor accompanied by IR projector, see Figure 3.3.1. The middle one is a simple RGB sensor, the one on the left is the IR projector and the one on the right is the IR sensor. The Primesense sensor is used in ASUS Xtion PRO. The IR projector projects a pattern of IR dots, see Figure 3.3.2a, which falls on all objects in the range of camera, the dots are detected using a conventional CMOS image sensor with an IR filter. The pattern will change size and position base on how far the objects are from the source. The density of the pattern changes with distance of the object. The depth sensor generates depth value for every pixel, as is shown in Figure 3.3.2b. The resolution of the depth image and RGB image has to be the same as the pixel information from both images are aligned together by the integrated image processor. The native resolution 640×480 is used in this thesis.

The application programming interface (API) for ASUS Xtion PRO is provided by Open Natural Interaction (OpenNI) [13] development kit. The OpenNI was found by PrimeSense and ASUS to provide open source API for 3D sensing and applications. It enables user to work with audio stream, images, tools for hand gestures and body motion tracking. The information about image is accessed through this interface, as its resolution, pixel format and pixel map of the image, etc.

Unfortunately the principle used in this camera does not allow close range performance, the minimum range is approximately 470 mm. The reason is that the projected pattern is so bright that the camera is not able to recognize it. And the depth sensor is not able to



(a) infrared image of the pattern



(b) depth Image

Figure 3.3.2: Infrared pattern project into the dept scene, image adapted from [1].

evaluate the data, so they are assigned to zero distance.

Whereas the robot is approaching the stairs in a close range, this disadvantage can cause a problem, as the robot is not able to recognize the stairs. This could be solved by installing the camera on rear side of the robot; however, the range of the camera will be limited as the legs are moving above the height of the body during the motion of the robot.

Another solution of this problem is installing the camera on the front side where the view is clear. Then, the parameters of the stairs can be calculated from a distance and the robot moves towards the stairs for the distance that equals the distance between the camera and the first stair.

Besides, there is still a problem considering the used principle of the camera with interaction with some surfaces, such as glass and surfaces that absorb the IR projection. The sensor is not able to evaluate these surfaces; so, the sensor treats them in the same way as close-range surfaces.

Chapter 4

Robot motion on planar surface

We can define a robot motion gait within its coordinate system with origin at the center of its body, see Figure 4.0.1. For motion on a planar surface, we assume the body is at the constant height above the surface and motion is done by particular motion of each leg within the coordinate system.

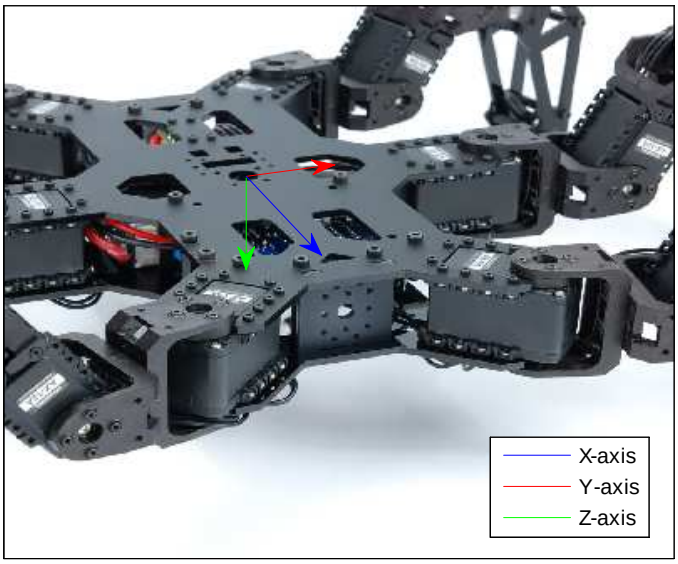


Figure 4.0.1: Coordinate system of the robot

The reason for defining the Cartesian coordinate system is to ease calculation of the motion. However, the control of the goal position of each actuator is done in spherical coordinates, so there has to be defined another set of coordinates for each leg. That means three angular coordinates has to be defined for one leg, each coordinate represents angular

position of one actuator as shown in Figure 3.1.1b.

4.1 Motion gaits

Regular periodical gaits for hexapod mobile robots can be classified by the number of legs supporting the body. There are three main groups of gaits: 1) 3+3 tripod gait that has three supporting legs, 2) 4+2 quadruped gait that has four supporting legs and 3) 5+1 one by one moving legs with five supporting legs [6]. These types of gaits are the only possible as there is no other combination of legs allowing movement..

Gaits differ from each other in stability during the movement, velocity of the gait that is connected with the number of steps in one cycle and robustness, in case of fault of leg.

4.1.1 3+3 tripod gait

The tripod gait is defined by having three legs supporting the body while three legs are lifting off and swinging forward. The number of steps in one cycle is two, which is the lowest number among gaits. That makes the tripod gait the fastest and most efficient gait; however, it is the least stable gait, as the robot is only supported by three legs. In case of fault in one leg, this gait is useless, as it is obligatory to have six functional legs for proper use of this gait.

The legs are divided into two groups of three, in a way that the three legs create a stable triangle. It means selecting front and rear leg on one side and middle leg on the other side, which makes the most stable triangle, as is shown in the first box in Figure 4.1.1.

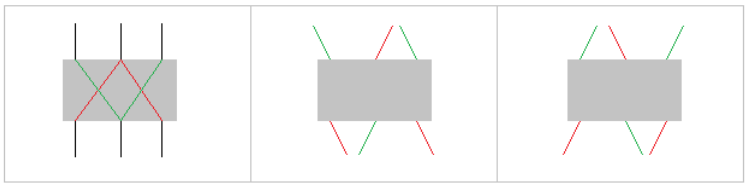


Figure 4.1.1: Tripod gait

4.1.2 4+2 quadruped gait

In this type of gait, the legs are divided into three groups of two. In each moment, two groups (four legs) are supporting the body, while two legs are swinging forward. Because of the body being supported by four legs, the robot is still stable even if there is a fault in one leg; however there is a condition that the fault leg is not the middle one, then the robot will fall.

Figure 4.1.2 depicts a possibility of quadruped gait with one pair swinging and other only supporting. The move forward is done at the end of cycle, thus, it slows down the robot. It can be solved by four legs supporting and pushing forward at the same time.

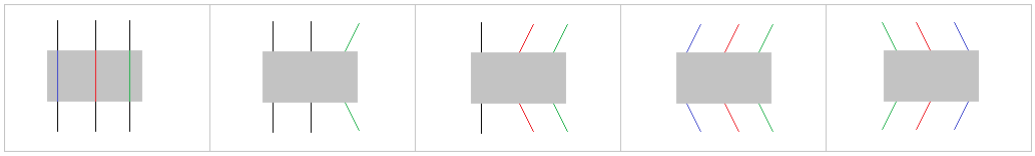


Figure 4.1.2: Quadruped gait

4.1.3 5+1 one by one gait

The last type of gait is defined by five legs supporting the body and one leg swinging forward. The legs can swing forward in any order, but generally it is done clockwise or counter clockwise. This gait provides the most stable form of motion, as it can move even if there is a fault in one leg; however, due to robust of the gait, it is also the slowest of all the gaits.

The three presented gaits are the basic regular periodical motion patterns designed for hexapod robots, they can be done various modification on them. The quadruped gait was chosen to be the starting point, at the beginning of designing the motion gait for stairs climbing. It is because the support is done by four legs that seemed to provide enough stability. Direct approach to stairs was supposed, so there could be done a motion control, which would treat both legs in pair in the same way.

While designing the motion, the quadruped gait provided insufficient stability, as the legs were standing close to each other. The solution was to change the gait and choose one by one gait. As it was the only one that provided more stability, despite the fact that it slowed down the climbing process.

4.1.4 Motion

The motion of the six-legged robot is done by gait in which the feet of the robot perform periodical movements. This movement is defined in the coordinate system of the robot. The position of each foot (x, y, z) cyclically return to initial position $(x_{init}, y_{init}, z_{init})$ defined before the motion of the robot. These values are defined in Cartesian coordinates with the center in each Coxa actuator. The basic motion is done by three elementary movements: 1) moving the leg up, 2) moving the leg down and 3) moving the leg backwards, which results in a movement of the body forward. The elementary motion of moving the leg backwards is usually divided into a number of steps that provides stability of the robot. It means when one leg is in the air, the rest remains on the ground and pushes the body forward. The calculating of the foot position is done by adding coordinates of vector of elementary movement to coordinates of the foot position. All the values being calculated are in millimeters.

The motion is defined by following parameters: vector of movement (x_v, y_v, z_v) that defines the direction of the movement, the number of steps in one cycle (s_{cycle}), the number of steps that pushes body forward (s_{push}), height of lift of the leg (l) and time of one cycle (t). The calculations of the particular movements in Cartesian coordinates are described in Table 4.1.

Table 4.1: Motion calculations

Movement	UP	DOWN	BACKWARDS
x_i	x_{init}	$\frac{x_v s_{push} t}{2 s_{cycle}} + x_{init}$	$x_{i-1} - \frac{x_v t}{2 s_{cycle}} + x_{init}$
y_i	y_{init}	$\frac{y_v s_{push} t}{2 s_{cycle}} + y_{init}$	$y_{i-1} - \frac{y_v t}{2 s_{cycle}} + y_{init}$
z_i	$z_{init} - l$	z_{init}	z_{init}

The results of the gait calculations are added to the initial position of the leg,

$$x = x_i + x_{init}$$

$$y = y_i + y_{init}$$

$$z = z_i + z_{init}$$

Each leg performs the same cycle, but each of them starts by a different movement. There are differences between each motion pattern. Each motion pattern is defined by the number of legs that are simultaneously in the air, the order in which the legs are lifted and the time that is necessary to perform one cycle. It is worth mentioning that motion pattern described in this thesis are on of many different motion patterns as it is very easy to define your own motion pattern.

4.2 Motion control

The position gained for each foot is in the Cartesian coordinates, but the goal position of the actuators is set in the joint coordinates. Thus, it is necessary to transform the Cartesian coordinates (x, y, z) into joint coordinates $(coxa, femur, tibia)$ using the inverse kinematic task (IKT), which provides the necessary calculations for the transformation.

Each actuator has a limitation on the possible goal position, which arises from construction of the actuator. The calculations of the motion does not include these limits, thus the system behaves as an open loop system. So when a goal position that is not inside the limit is requested and send to the actuator,, the algorithm only informs that a goal position is not being send to the actuator and skips setting of the goal position for the next actuator.

The joint coordinates, provided by IKT are the same for each leg of the robot; however, the construction of left and right legs is different and also the construction of front, middle and rear leg is not the same. Thus, there has to be made a transformation that would modify the joint coordinates into values appropriate for each actuator. The transformation for each actuator is presented in Table 4.2. As the Coxa actuators are not attached to the body at the same positions, there is an angle

$$\alpha = 42.2^\circ,$$

that represents the rotation between the rear, front and middle Coxa actuators. Femur and Tibia actuators are also rotated as the connection between the actuators is not direct and plastic brackets are used. For the Femur actuator, is applied a rotation about the angleβ and for Tibia rotation aboutγ angle.

$$\beta = 3.5^{\circ}$$

$$\gamma = 46.3^{\circ}$$

Table 4.2: Transformation for particular actuator

Leg	Coxa	Femur	Tibia
Right front	$-\alpha$+*coxa*	β+$femur$	$-\gamma$+*tibia*
Right rear	$\alpha - coxa$	β+$femur$	$-\gamma$+*tibia*
Left front	α-*coxa*	$-\beta$-$femur$	γ-*tibia*
Left rear	$-\alpha + coxa$	$-\beta$-$femur$	γ-*tibia*
Right middle	*coxa*	$\beta + femur$	$-\gamma$+*tibia*
Left middle	*coxa*	$-\beta - femur$	γ-*tibia*

When the final position of the actuator for the next step is calculated, the actuator is not set to this position immediately, but it is approaching the final position in small moves to provide a smooth motion. Actuators do not change their speed of movement, but the moves are set in a way that all actuators approach the goal position at the same time. When all actuators approach their goal positions, the next step is calculated.

The goal position necessary for setting the actuator can be sent by eighteen messages, which are easy to create, but it takes some time, because all actuators have to be gradually set. The other way of setting the actuators is by using synchronous write, which is more complicated to assemble, but faster, because it sets all actuators with one message; however it does not provide the feedback from the actuators, as the communication is the broadcast. Another problem arising from the synchronous write is an immediate high current requested by the actuators operating simultaneously at the same time..

Chapter 5

Model of the robot motion for climbing on stairs

The original idea was to use the 4+2 quadruped gait for traversing the stairs, because in the type of motion shown in Figure 4.1.2 it was possible to use the full range of Coxa actuators. As the swinging of the legs was done from front to rear, there was not a possibility of the legs hitting each other; thus, full range of the Coxa actuators can be used. However, later, it came up that this type of gait does not provide enough stability for the robot, that is why the 5+1 gait was chosen for the stairs climbing algorithm, because it provides better stability. The swinging order from the front to the rear legs appears to be useful. The order is shown in Figure 5.0.1.

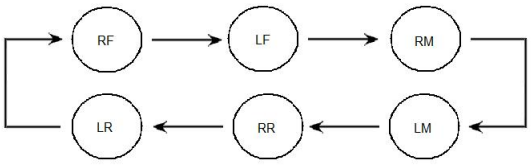


Figure 5.0.1: Move order of legs of the robot in the 5+1 gait

The proposed strategy to climb the stairs can be divided into three phases:

1. It is assumed that the robot is moving on a planar surface and there are parameters that differ when being used during the motion on planar surface and during climbing on the stairs. One of them is the initial position of each leg; the second one is the different motion gait used on a planar surface and on the stairs. The first phase is considered to create a smooth transfer between the motion on a planar surface and climbing on the stairs.

2. When being prepared for stair climbing, the motion changes into regular periodical gait that last until the robot reaches the top stair.

3. After reaching the top stair, the robot has to change the gait and the initial position of each leg back for motion on planar surface.

5.1 Stair climbing gait

There are two ways of climbing the stairs considering the robot body. It can either rotate the body in such a way the pitch of the body will be identical to the slope of the stairs, or let the pitch be in the zero angle. The second way was chosen in this approach, as it provides more stability during the movement.

When designing the motion for stair climbing, it is necessary to find limitations of the motion of the robot; however, it is not possible without knowledge of the foot motion in the operational space. Because the motion with no body rotation was chosen, it is possible to estimate the operational space of each leg. As it would be difficult to design a motion within the entire operational space of the leg, and therefore, we simplify the problem and made the value at the Y axis constant. Based on the experimental experimental measuring we establish the value as (in millimeters)

$$y_{init} = \begin{cases} 130 & Right\,leg \\ -130 & Left\,leg \end{cases}.$$

Then the foot is not too far to lose grip, and therefore it still properly supports the body, and it is also not too close to cause the operational space to be tight.

By choosing the 5+1 one by one gait it is possible to estimate the number of steps in one cycle,

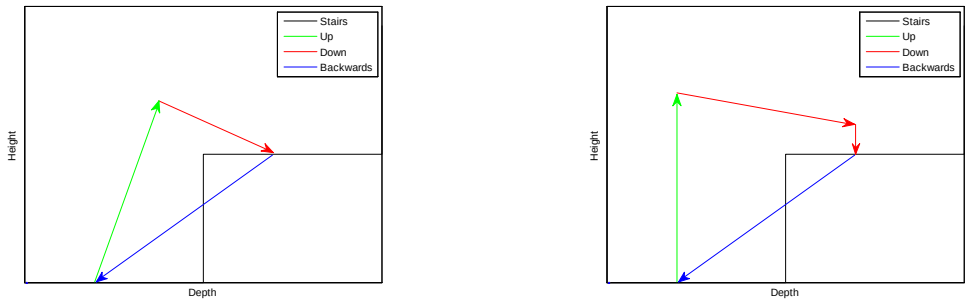
$$s_{cycle} = 6a = 18,$$

where $a = 3$ is number of movements in the air (swinging the leg forward), and also estimate the number of push steps, which pushes the body forward, so,

$$s_{push} = s_{cycle} - a = 15.$$

The stair climbing gait is based on the motion on planar surface, so it originally uses the same principle of leg movement. However, there is a problem while moving the leg down, which did not appear in motion on planar surface. The problem is that the move down can be so close to the stair that the leg can accidentally rub the stair and thus push the robot backwards. Therefor, the move down has been changed so that it could properly move and land the foot on the stair as it is shown in Figure 5.1.1.

For an appropriate functionality of the motion on the stairs, it is necessary to define an exact position of the initial position of each leg. Resolving this problem was not a trivial task, as there are more factors to be taken into account. It has to be defined, whether the initial positions will change according to different depth of the stairs. A question of how exactly will the legs move is tightly connected to this topic. Solving this task was more of an experimental than a theoretical work.



(a) original leg movement

(b) proposed final leg movement

Figure 5.1.1: Leg movement for stair climbing, where the movement of the leg up is marked with green color; the movement down with red color and the movement of the leg backwards is marked with blue color.

As there are three phases of movement on the stairs, the second phase was used to figure out the initial positions, because it is the phase in which the robot will spent the most time by traversing all the stairs.

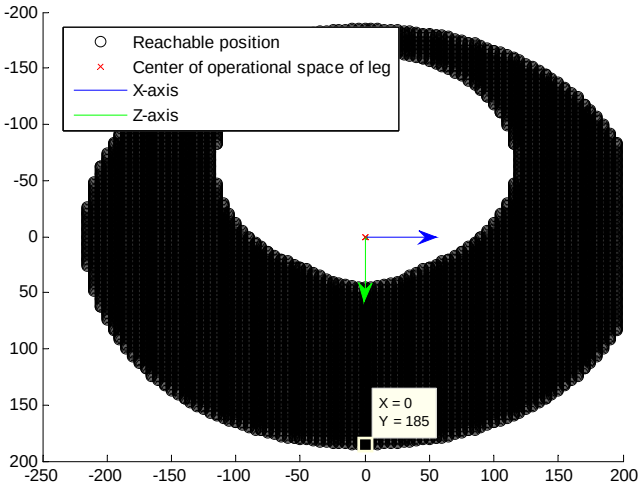


Figure 5.1.2: Operational space of front leg in XZ plane, where the reachable positions for the leg are marked with black. The center represents the position of Coxa actuator.

First idea was to set each pair of legs on one stair and change the distance between each pair according to the depth of the stairs. This solution came into problems, because the height distance in this type of positioning between front and rear leg can be two times longer than the height of the stair and as the motion is done by 5+1 gait, while moving forward, the rear legs have to move down in a distance that equals the height of the stair.

This demands an operational space of the leg being at least three times higher than the stair height. This results in a position of the middle and rear pair being close to each other, so the legs will occupy the least space on a stair.

Because the position on Y-axis was made constant, it is possible to picture the operational space of each leg in XZ plane. An operational space of the front leg is shown in Figure 5.1.2. The position of the Coxa actuator is represented as the center of the operational space.

As the synchronous movement of all legs of the robot is considered, the parameters (h,d) of the stair, used in the calculations, have to be modified, because while one leg is swinging forward, other legs move the body forward and thus changing the origin of the swinging leg. Using the real size parameters of the stairs leads to a behavior presented in Figure 5.1.3a, where the leg does not land in the appropriate position, but remains in the air above the stair. As the steep of the stairs has to remain the same in the robot model, both parameters has to be modified by an equivalent fraction f, as the real position is farther than it is supposed to be. The value of the fraction derives from the movement, as the movement consists of 18 steps, out of which 3 steps are needed for swinging the leg forward. As a result, the leg is being moved closer to the landing position for a distance of

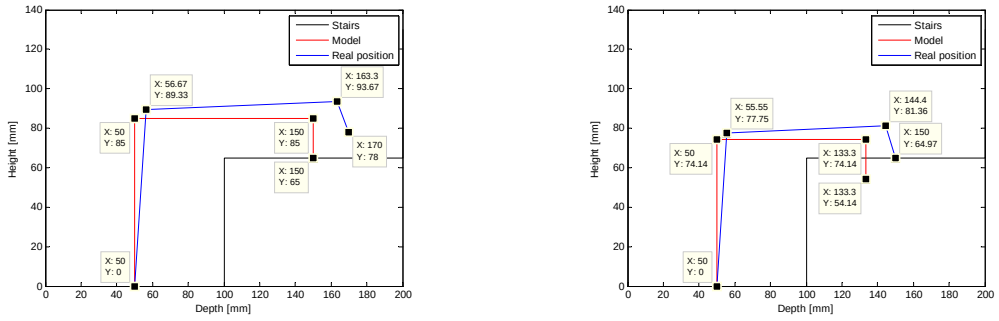
$$f = \frac{s_{push}}{s_{cycle}},$$

modifying the original parameters of the stair into,

$$\hat{h} = fh$$

$$\hat{d} = fd,$$

which leads to movement presented in Figure 5.1.3b.



(a) $f = 1$ (b) $f = {}^{15}/_{18}$

Figure 5.1.3: Model and real position of the leg

As the number of cycles s_{cycle} in the second phase is equal to number of stairs N, there has to be added steps to cover the third phase. The estimation of the number of steps necessary for accomplishing the first phase is discussed in Section 5.3.1. The phase three

can be performed in two steps; so, for now the combination of the phase two and three will be explained and

$$\tilde{N} = N + 2.$$

Parameter $\tilde{h}$ has to have two possible values, either the height of the stair, or zero, to land the foot on the same level. This parameter will differ for front pair,

$$\tilde{h}_i = \begin{cases} \hat{h} & i = \langle 0; \tilde{N} - 1 \rangle \\ 0 & i = \tilde{N} \end{cases}$$

and for the middle and rear pair

$$\tilde{h}_i = \begin{cases} \hat{h} & i = \langle 1; \tilde{N} \rangle \\ 0 & i = 0 \end{cases}.$$

In addition, during the move backwards in the second phase, the body moves under the same angle as is the steep of the stairs; however, during the phase one and two, the body moves only forward; so, there is need for variable h_F that is the same for all legs in each move backwards,

$$h_F = \begin{cases} \hat{h} & i = \langle 1; \tilde{N} \rangle \\ 0 & i = 0 \end{cases}.$$

The movement of the front pair is situated closer to Coxa actuators than middle and rear pairs; so, during the up movement in second phase, the foot of the leg cannot reach the appropriate position. But, the limitation considering the y_{init} parameter does not apply while the leg is in the air, as there is not need to support the body. That is why the last parameter $\tilde{y}$ applies only for up movement of the front pair of legs,

$$\tilde{y}_{init} = \begin{cases} 40 & Right\ front\ leg \\ -40 & Left\ front\ leg \\ 0 & Other\ legs \end{cases}.$$

During the up movement, the foot has to be moved over the stair, in order not to hit it, so the variable h has to be increased. The safe distance from the stair is considered to be 20 mm; so,

$$\bar{h} = \hat{h} + 20.$$

Having all the necessary parameters, the calculations for stair climbing gait are presented in Table 5.1. Calculations for the robot motion on stairs vary from those used on planar surface, as the real parameters of the stairs are used.

After calculation are done, the same transformation as in the planar surface gait is applied,

Table 5.1: Motion calculations for stair movement

Movement	UP	DOWN 1	DOWN 2	BACKWARDS
x_i	x_{i-1}	$x_{i-1}+\hat{d}$	x_{i-1}	$x_{i-1}-\frac{\hat{d}}{s_{push}}$
y_i	$\tilde{y}_{init}$	0	0	0
z_i	$z_{i-1}-\bar{h}$	$z_{i-1}+(\bar{h}-\tilde{h}_i)$	$z_{i-1}+20$	$z_{i-1}+\frac{h_F}{s_{push}}$

$$x = x_i + x_{init}$$

$$y = y_i + y_{init}$$

$$z = z_i + z_{init}.$$

5.2 Limitations

Regarding the stability of the robot, it is better to move the legs near the center of gravity of the body, otherwise they will not provide enough support. It can be seen in Figure 5.1.2 that the highest point of the operational space is right under the Coxa actuator in 180 mm, but the lowest point of the leg position on Z-axis, for the robot to be able to walk, is 60 mm. Otherwise, the foot of the leg will be nearly at the same height as the body, thus, the foot will not touch the ground. This leaves the total height of 120 mm for the robot movement and as the operational space has to be three times higher than the height of the stair. The maximum height of the stair is then 40 mm, only for the best position. That means approximately 30 mm as an average maximum height for different depth of stairs.

It was possible to increase the maximum height of the stairs by resetting the initial position of the legs. Instead of letting each pair being on one stair, it was changed in a way that front pair was on one stair, middle and rear pair on the second (lower) stair. This set up let the average maximum of the stair be approximately 50 mm.

However, it leads to a limitation in the depth minimum of the stair, as two pairs are being on one stair and the area necessary for two legs in a row is 90 mm. That means the minimum value of the depth of the stair is 110 mm, as it is needed to have a reserve in front of the pair and behind it.

As the upper limit of the depth of the stair has to be maximal, the initial positions at the X axis remain constant for any value of parameters (h,d), which provides the maximal operational space for the leg movement. The position of the front pair is at the level of the front side of the robot giving enough support for the front part of the robot during the movement. Middle and rear pairs are set in the most further position that holds up against falling backwards. Also middle and rear pairs are put as close as possible to each other in order to take the least possible space on the stair. Out of experimental measurements

came out exact values for the initial positions of each leg to provide maximal possibilities for the movement of the robot. Results for X and Y axis are presented in Table 5.2. It was also possible to measure the maximum depth of the stair to be 200 mm. Again, all the coordinates are considered with the origin of the particular Coxa actuator.

Table 5.2: XY axis values for initial position of each leg

Axis	RF	LF	RM	LM	RR	LR
X	-10	-10	0	0	50	50
Y	130	-130	130	-130	130	-130

To maximize the upper limit of the height of the stairs, it was necessary to make the initial position at the Z axis variable. The problematic part of the operational space of the leg is for the parameter d reaching one of its limits and the parameter h is set to its upper limit. In this case, the body has to be lowered in order to compensate the movement performed at the edge of the operational space, see Figure 5.2.1. The value on the Z axis remains the same for each leg, as the body has to be kept in the horizontal position. This increases the final maximum height of the stairs to 60 mm.

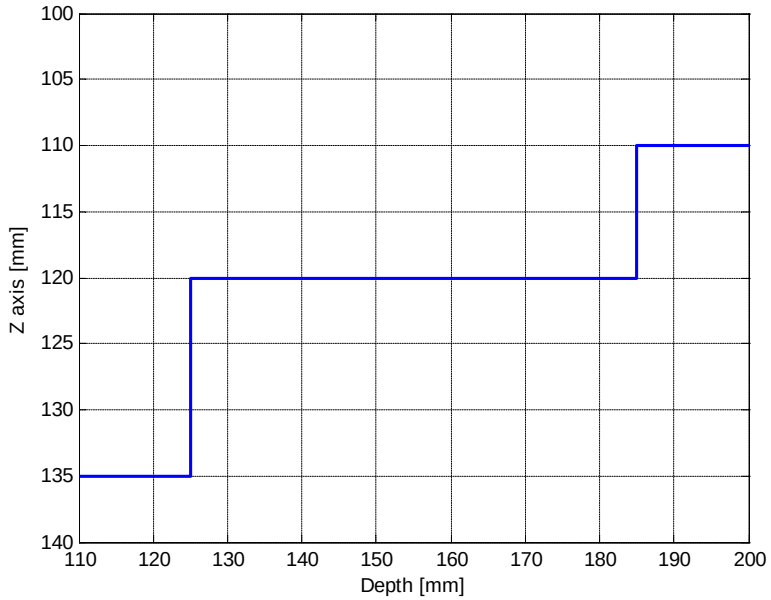


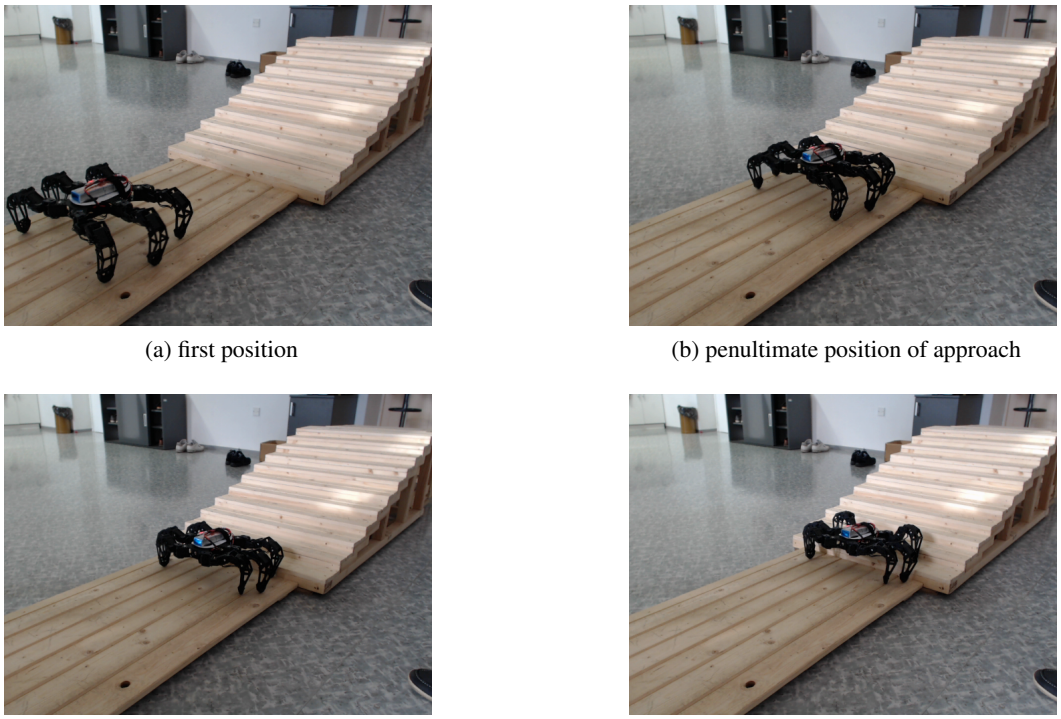
Figure 5.2.1: Dependency of Z axis value to depth of the stair

5.3 Movement

As the movement is considered to be of three different phases, it is necessary to put all of them together using the same gait; otherwise, the stair climbing algorithm used in second phase is useless as the robot would not able to accomplish the first and third phases.

During the second phase of the stair climbing, all the legs perform movements with same parameters (h,d). In the first and third phases, the parameters vary for each leg, as they are not performing the same movement. These parameters depend on the position of the robot at the stairs.

5.3.1 First phase



(a) first position

(b) penultimate position of approach

(c) reaching the frontier of the stairs

(d) last position of the first phase

Figure 5.3.1: First phase of the movement

When the robot is placed in front of the stairs, it gains the parameters using the depth camera; however, it has to be placed at the specific distance to allow capturing a depth image of the scene. The exact distance from the position of the camera, on top of the robot, to stairs, in front of it, can be measured by the depth camera. The first movement of the robot in the first phase is change of the initial position of each leg from initial positions for planar surface into initial positions for the stair climbing; however, the rear legs are placed more to the back to gain more stability, as there is no limitation in leg placement during the

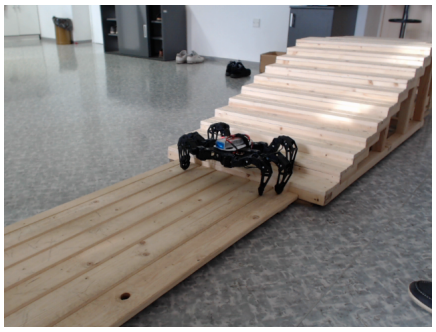
first phase. The position of the rear pair switches to the initial position, as designed for stair climbing algorithm, right under the stairs, i.e., during the penultimate step of the first phase. The motion during the approach is done by the same algorithm as for the stair climbing and the step is the movement performed during one cycle. Regarding the parameters of the step, the value depth of the stair is treated like the parameter d in the stair climbing algorithm and the value of the height is treated like the parameter h.

The depth of the step that the robot makes in order to move itself right under the stairs is a constant of 180 mm. This value is a compromise between a fast movement and stability. It is calculated how many steps the robot has to make in order to get to stairs as close as possible and then make one smaller step that equals to the rest of the distance to get right in front of the first stair. When the robot is right in front of the stairs, the depth of the step is set to depth of the stair d and the last step in first phase is performed.

The height of each step equals to zero, as the robot is moving on a planar surface in front of the stair. The height of the step is set to a nonzero value, when the robot is standing in front of the stairs and has to put the front pair on the first stair, while the rest of the legs remain on the planar surface. The height of the step changes only for front legs, as they are the only one that has to be moved up. In this case, the value of the real height of the stair h is applied, skipping the modification into $\hat{h}$.

During the first phase, the body remains in the same height; so, parameter h_F is set to zero for the whole first phase. As the end position of the robot in the first phase is designed to simultaneously be the first position in second phase, the motion of the robot can be fluently transferred to second phase. Important positions of the first phase are depicted in Figure 5.3.1a, where it represents the starting position of the robot, Figure 5.3.1b represents the last position of the robot using initial positions for approach; in Figure 5.3.1c, the reached the exact frontier of the stairs and initial positions are set to initial positions for the stair climbing; Figure 5.3.1d shows the last step of the first phase, when the robot is ready to initiate the climb towards the top of the stairs.

5.3.2 Second phase



(a) first position of the second phase

(b) last position of the second phase

Figure 5.3.2: Second phase of the movement

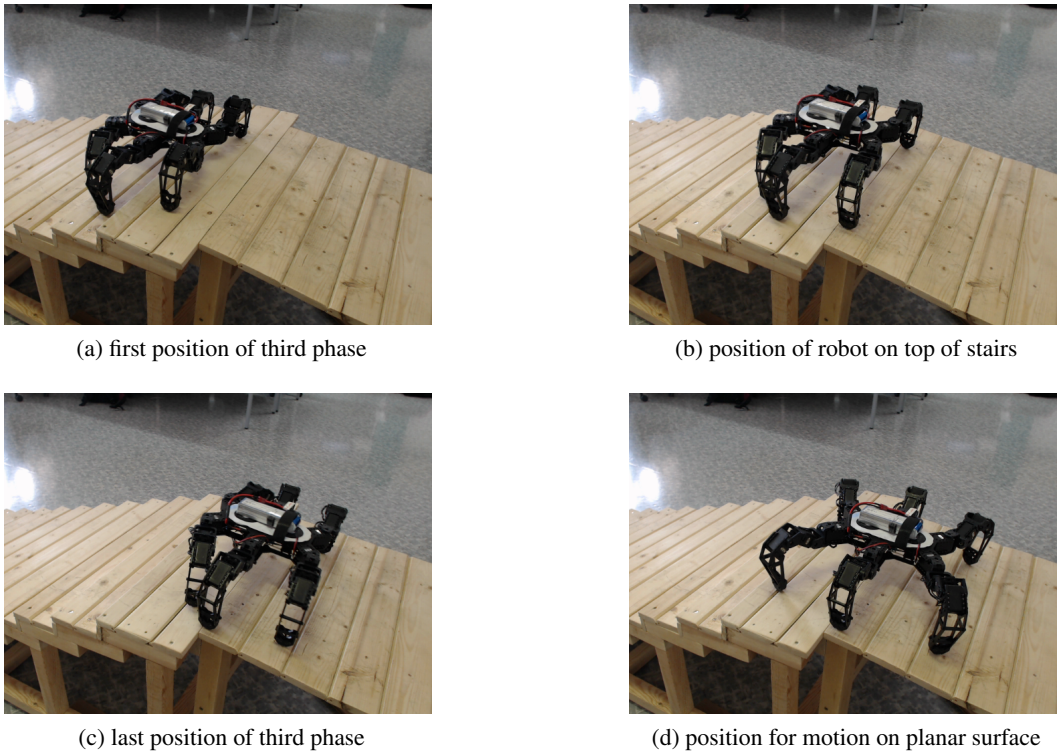
After finishing the first phase, the robot is situated in a position from which it starts the climbing. Each leg repeatedly comes back to this position after performing one cycle. The robot is aware of its initiate position, of the moves that it has to commit in order to climb one stair and the number of stairs it has to traverse, which gives it all the information necessary to reach the top of the stairs. During the second phase, at each time, the robot knows on which particular stair it stands.

The depth of the step is calculated according to the algorithm for the stair climbing and it is set to d. This parameter does not vary during the second phase, neither does the parameter for height of the step, which is set to h.

As the body is moving up the stairs, it remains in the horizontal position. To keep the body moving under the same angle as is the steep of the stairs, the parameter h_F equals to the parameter $\hat{h}$. So, the body is shifted upwards for the height of the stair, as the shift is done through the whole cycle.

The second phase of the movement consists of repetitive movements of each leg forcing the robot to climb the stairs. It finishes, when the front legs of the robot are placed on the last stair and the remaining four legs are placed on the penultimate stair.

5.3.3 Third phase



(a) first position of third phase

(b) position of robot on top of stairs

(c) last position of third phase

(d) position for motion on planar surface

Figure 5.3.3: Third phase of the movement

The third phase of the movement represents the final phase and it is considered to transform the stair climbing motion back into a motion on a planar surface. It has to be able to fluently continue after the second phase, so, the third phase starts with front legs situated on the last stair and the other four legs situated on the penultimate stair.

The depth of the step remains the same as in the second phase, i.e. d. The height of the stair has to vary for the front legs and for the remaining four legs. The front legs are already on the top of the stairs, so there is no need to move them higher; thus, the height of the step for the front legs is set to zero for the whole third phase. But the remaining four legs are standing on the penultimate stair, so, there has to made one step with the height that equals h in order to get the four legs on top of the stairs. Then, the value has to be changed to zero, because all the legs are then placed on the top of the stairs. The variable h_F equals $\hat{h}$, as the body has to be lifted high enough for the robot to be able to move on the upper platform of the stairs.

But the robot is still placed with its rear legs on the edge of the last stair. So, it has to move for one more step in order to safely change the initial positions of each leg into positions for the planar surface motion. One more step is added to the movements to finalize the stair climbing.

During the last step, the both variables h_F and the height of the step is set to zero as the robot is only moving forward without traversing any obstacle. The depth of the stair still equals to d, as the only purpose of this act is to move the robot into safe distance from the edge and the minimum depth of the stair is 110 mm, which is the safe distance for its change.

At the end of the third phase, the initial position of each leg of the robot is set to the initial positions for a planar surface and the robot is able to continue with other movements.

Chapter 6

Camera

The necessary parameters for stair climbing are height of the stair, depth of the stair, width of the stair, number of the stairs and distance of the robot from the stairs. These parameters are gained from the depth camera ASUS Xtion Motion.

It is necessary to respect limitations of the used RGB-D camera to capture a depth image of the stairs for detection of the stairs parameters. First, the robot has to be placed farther than 60 cm from the stairs, because of the principle used by depth cameras with Primesense sensor. Second, it is assumed that there is nothing else than the stairs in front of the robot for a reliable processing the image data.

The picture of the stairs taken by the depth camera is presented in the Figure 6.0.1, where dark red pixels represent closer points and light red pixels represents farther points.

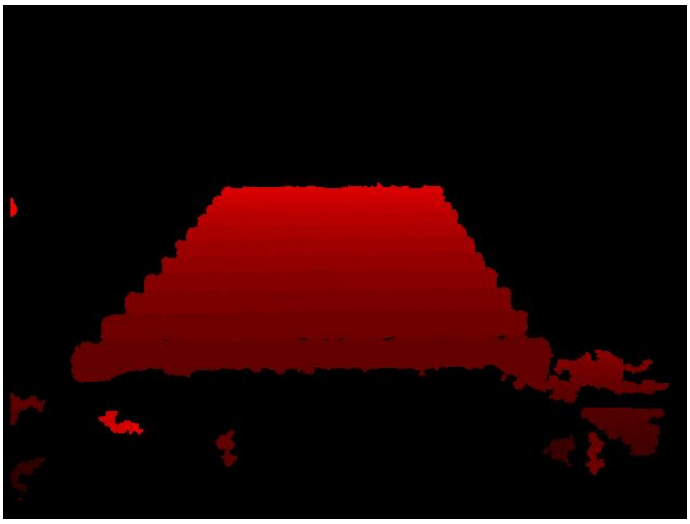


Figure 6.0.1: Depth picture of the stairs

Considering the necessary parameters and representation of the stairs in image gained from the camera, the first parameter to estimate is the depth of the stair.

6.1 Depth of the stair

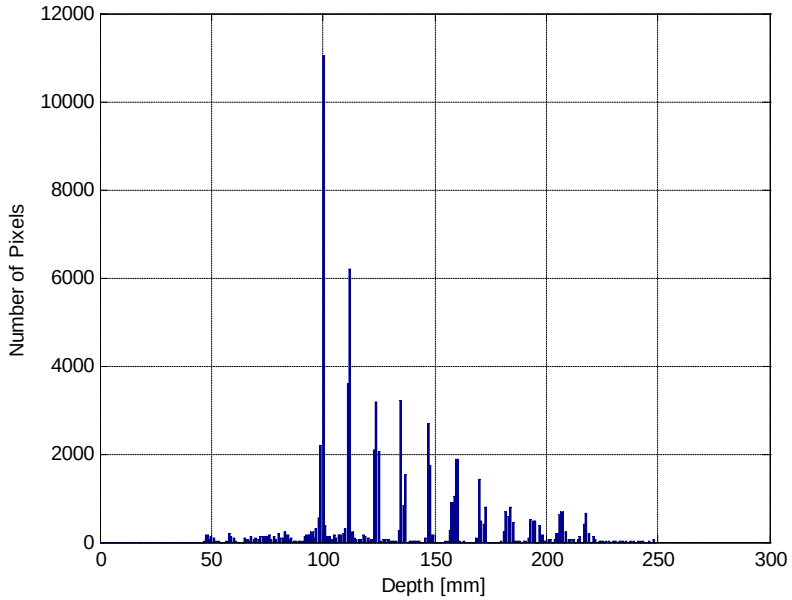


Figure 6.1.1: Histogram of the depth pixels

The depth of the stairs is the first gained parameter. Each pixel in the image carries a value of its distance from the camera, and as it is supposed there is nothing else in front of the camera except stairs. Thus a histogram of the depth data can be created, where the steps will appear as the peaks. However, the values for each pixel is represented in millimeters, so, the peaks are not significant. The histogram is improved by gathering the values into groups of centimeters, which covers larger area; thus, it is easier to look for peaks, but it is still possible to gain the distance of each step from the values. Figure 6.1.1 represents a histogram of the different depths from 0 to 300 cm, where each peak represents one stair. The top of each peak is decreasing with distance of the step from the depth camera, as objects that are further covers less space in the image.

It can be seen in the image that there is a noise that occurs among the depth values. The peaks that are closer to the camera are significant against the noise and they occupy a narrow space, in contrast to the peaks that are farther. It was experimentally measured that the first five steps can be positively recognized in the image.

The first five stairs are being used for gaining the depth parameter. Despite being positively recognized, there is still a correction of the image that has to be done. In a case the area around the peak is not narrow and the surrounding values are reaching closer to the peak value, the average value of the surrounding is being taken as the position of the particular stair p_i. The position is relative to the position of the camera and represents the distance of each stair from the camera. The position of the first five steps are being taken

and depth value is calculated according to

$$d = \frac{\sum_{i=2}^{5}(p_i - p_{i-1})}{4}.$$

6.2 Distance to the stairs

The approach from position of taking the image of the stairs to the position right at the stairs is a part of the stair climbing algorithm, the robot has to be informed about the distance. For gaining the distance to stairs d_r, the same histogram presented in Figure 6.1.1 is used. If the position p_1 of the first stair is positively recognized in the image, it represents the distance from the camera to the first stair.

The camera is placed on the front side of the robot, at the same level as the front legs of the robot, so the distance from the camera to the first stair equals the distance from the front legs to the first stair.

$$d_r = p_1$$

6.3 Number of the stair

(a) image before filter (b) image after filtering

Figure 6.3.1: Pixels representing the second stair

It is assumed that the peaks shown in Figure 6.1.1, represent the positions of the stairs. This applies to the first five steps; however, when there are more stairs to traverse, the identification of the real position of the stairs is complicated, as the pixels, which represent the stair, are stretched to a wider area, and the peak is formed by a lower value.

The depth of the stair is used to deal with these complications, as the position of the first step is a known value, the position of the other stairs can be estimated, because it is assumed that all the stairs are identical. The estimation is checked in a way that the sum of pixels in the surrounding is compared to a threshold. If the sum decreases below the threshold it

represents the end of the stairs. The values after the last peak drop nearly to zero, as can be seen in Figure 6.1.1, and they can be positively distinguished from the surroundings of the estimated peak values.

6.4 Height of the stair

For gaining the value of height of the stairs, it is necessary to acquire a clear shape of one stair from the image, as all the stairs are identical. A closer stair to the camera appears in the image with a better shape. Due to problems regarding the construction of the first stair (a gap under the stair), the second stair was chosen for estimation of the height parameter.

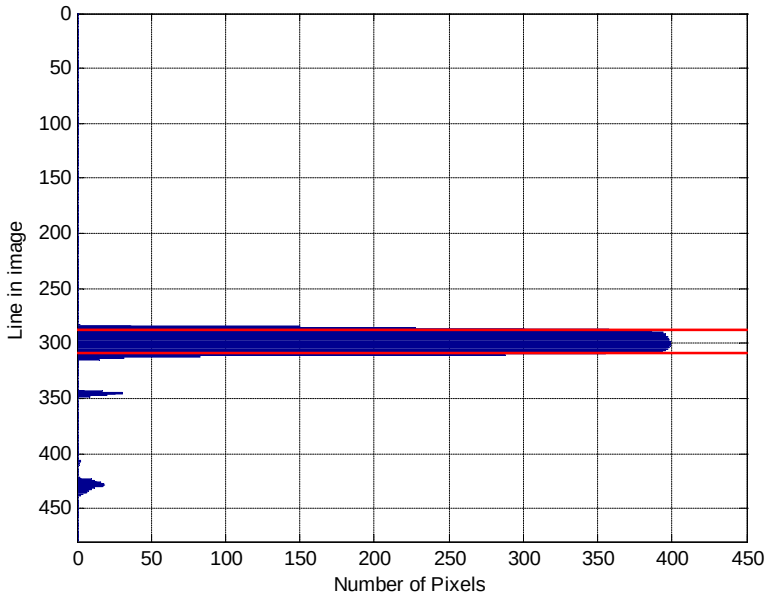


Figure 6.4.1: Histogram of the number of pixels per line in the image, shown in Figure 6.3.1b, where l_{start} and l_{end} are marked by red line

The position of the second stair p_2 is known from Section 6.1; thus, it is straightforward to filter the pixels representing the second stair. The result of filter is shown in Figure 6.3.1b. The filter is applied to all pixels that are within two centimeters distance from the position p_2.

The camera is taking a image of the stair in a way that it creates a rectangle in a horizontal position, because the robot is standing straight in front of the stairs. The image of the stairs has resolution of 640×480, so there are 480 lines of pixels. The wanted rectangle is placed horizontally, so it should appear in the histogram of number of pixels per line, see Figure 6.4.1, as a group of significant values. The values, that identify the height of the stair, are a line in which the rectangular starts and a line in which the rectangular ends, both in horizontal direction. By comparing the values of each image to a threshold, the start line of

the rectangle l_{start} is found, which is set to 350 pixels per line. The value was chosen high enough to deal with the noise, but at the same time low enough to allow the identification. After identification of the rectangle start, the last line l_{end} is found, when the value drops under the threshold.

6.5 Width of the stair

The width of the stair w is needed, to decide, whether the robot fits on the stairs. The approach to identify w is similar to estimation of the height. It is based on the same image shown in Figure 6.3.1b, but a histogram of the number of pixels in a particular column is created. However, due to the fact that the stair is represented in the image as a rectangle, the noise can behave like a very thin narrow stair; so, the identification is not as simple as in Section 6.4, where the height is represented by a high number of pixels associated to a small area. The threshold value in this section is a result of experimental measurements and the final value is 15 pixels per column. The histogram of the number of pixels per line is presented in Figure 6.5.1.

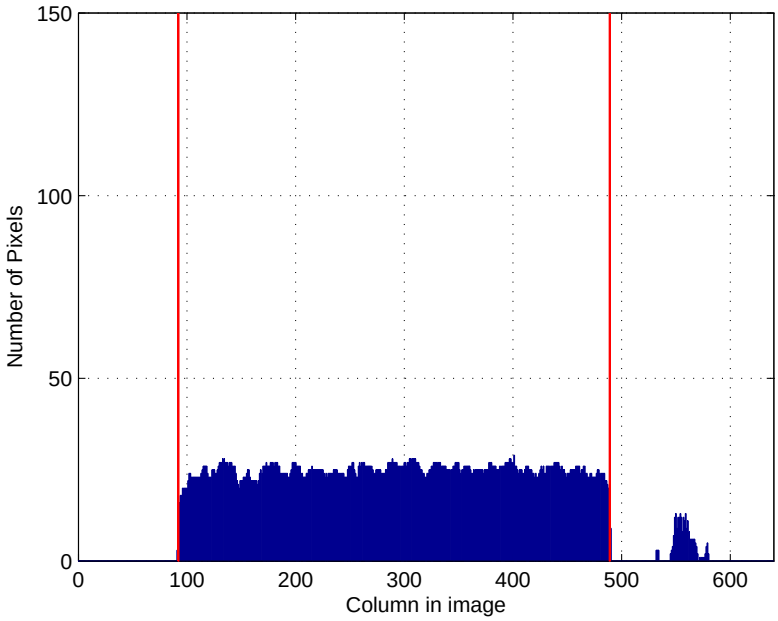


Figure 6.5.1: Histogram of the number of pixels per column in the image, shown in Figure 6.3.1b, where c_{start} and c_{end} are marked by red line

The width of the stair is defined by the pixel distance in the image, as the rectangular shape is placed horizontally and bounded from both sides by columns. The number of the pixels in each line is compared to the threshold and the start column of the rectangle in the vertical direction c_{start} is found. Then, the values of each column are again compared to the

threshold and when the value drops under it, the end column c_{end} is found.

The width of the stair w has to fulfill the condition Equation 2.1.1.

6.6 Pinhole camera model

The values of height and width of the stair are estimated in the image coordinates, therefore, a transformation to the world coordinates is needed. The pinhole camera model [14] is used to define the relationship between the image size and the real world size, called the perspective projection. The center of the perspective projection is denoted as the optical center and the line perpendicular to the image plane passing through the optical center as the optical axis. The intersection point of the image plane with the optical axis is called the principal point. Assuming a camera with the optical axis being collinear with the Z axis and the optical center being located at the origin of the 3D coordinate system, the situation is presented in Figure 6.6.1.

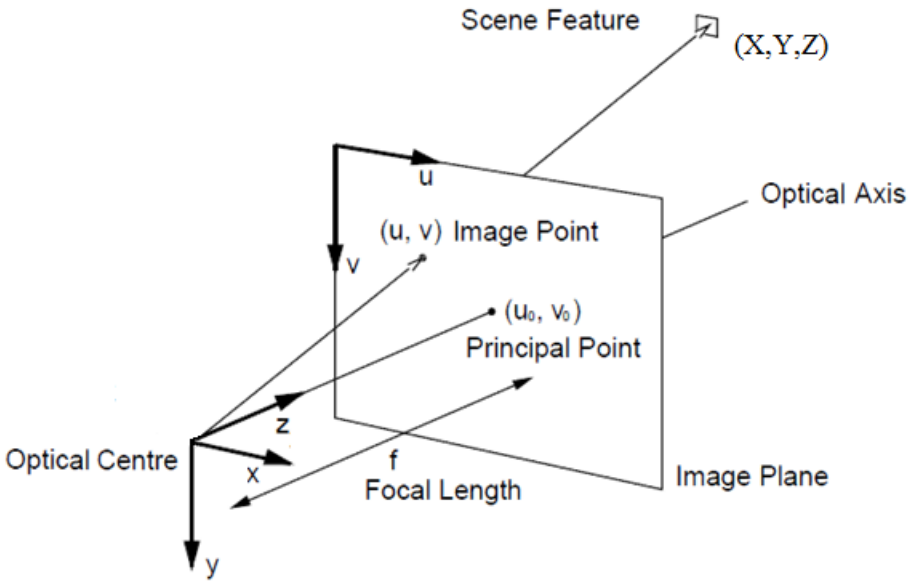


Figure 6.6.1: Pinhole camera model

The transformation of the point in 3D $(X,Y,Z)^T$ onto the image plane at the image point $(u,v)^T$ can be written as,

$$u = \frac{Xf}{Z}$$

$$v = \frac{Yf}{Z},$$

where f denotes the focal length of the camera. The values for the focal length of the camera were taken from the Computer Vision Group of the Munich Technical University [15]. The

group was working with a Kinect, estimating all the intrinsic parameters for the IR camera. As the Kinect camera uses the same Primesense sensor as ASUS Xtion Pro, it was possible to use the provided results of the calibration of the infrared camera. The relations can be formulated as

$$(\lambda u, \lambda v, \lambda)^T = (Xf, Yf, Z)^T,$$

which can be expressed in a matrix notation

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix},$$

where $\lambda = Z$ is the homogenous scaling factor and the value Z is estimated from the position of the second step p_2:

$$Z = p_2.$$

The pinhole camera model has the origin of the pixel coordinate system at the principal point $(u_0, v_0)^T$ located at the center of the image; however, the image taken by the camera has the origin of the pixel coordinate at the top-left pixel of the image, thus, a conversion of the coordinate system is necessary. Using the homogenous coordinates, the position of the principal point can be integrated into the projection matrix. The perspective projection then becomes

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}.$$

A transformation from the image plane to the point in 3D can be formulated as follows:

$$X = \frac{\lambda u - u_0 Z}{f},$$

$$Y = \frac{\lambda v - v_0 Z}{f}.$$

To estimate the real parameter of the depth and width, the parameters described in Sections 6.4 and 6.5 are used:

$$\begin{aligned} h &= \frac{\lambda l_{end} - v_0 Z}{f} - \frac{\lambda l_{start} - v_0 Z}{f} = \frac{\lambda (l_{end} - l_{start})}{f} \\ w &= \frac{\lambda c_{end} - u_0 Z}{f} - \frac{\lambda c_{start} - u_0 Z}{f} = \frac{\lambda (c_{end} - c_{start})}{f} \end{aligned}$$

Chapter 7

Experimental results

The proposed motion gait for climbing stairs has been experimentally verified in set of setup. First, exact model of the stairs has been used and the robot was almost always able to climb the testing stairs. After that, the gait has been verified with automation estimation of the stairs parameters. The validation has been performed using stairs with the parameters:

Height of the stair: 40 mm;

Depth of the stair: 120 mm;

Number of the stairs: 11.

The measuring was done in four different distances (680 mm, 750 mm, 820 mm and 865 mm) from the stairs, to verify robustness of the estimation of the parameters. At each distance, several measurements have been performed. Each measurement is presented in Tables 7.1 to 7.4.

Table 7.1: Stairs estimation of d_r = 680 mm

No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Error [%]**
d	118	118	118	118	118	118	118	118	118	118	1.67
h	40	40	40	40	40	40	40	40	40	42	1.58
N	11	11	11	11	11	11	11	11	11	11	0.00
d_r	680	680	680	680	680	680	680	675	680	680	0.23

Table 7.2: Stairs estimation of d_r = 750 mm

No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Error [%]**
d	115	117	117	117	117	117	117	115	117	117	2.91
h	42	42	42	42	42	42	42	42	41	42	4.80
N	11	11	11	11	11	11	11	11	11	11	0.00
d_r	750	750	750	750	750	750	750	750	750	750	0.00

Table 7.3: Stairs estimation of d_r = 820 mm

No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Error [%]**
d	117	117	117	117	117	117	117	117	117	117	2.50
h	41	39	39	41	41	41	39	39	39	41	2.5
N	11	11	11	11	11	11	11	11	11	11	0.00
d_r	820	820	820	820	820	820	820	820	820	820	0.00

Table 7.4: Stairs estimation of d_r = 865 mm

No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Error [%]**
d	118	118	118	118	118	118	118	118	118	118	1.67
h	41	41	41	41	40	41	41	43	41	41	3.26
N	11	11	11	11	11	11	11	11	11	11	0.00
d_r	865	865	865	865	865	865	865	865	865	865	0.00

The number of steps remained the same in all four measurements; thus, this estimation can be considered stable and robust.. There should be no problem during the stair climb with this parameter.

The distance of the camera from the stairs is the second most stable value, as it varied only once from the exact value. There is no problem in slightly exceeding the exact value, as the robot will only push itself from the first stair, remaining on all six legs, without losing stability, and as a result the robot will be standing right at the beginning of the first step. On the other side, if the robot will not end its approach towards the stair at the beginning of the first stair, it may encounter a loss of space at the end of the climb, as there would not be enough reserved space. Being right at the beginning of the stairs increases tolerable error in parametrization of the stair depth.

The height of the stair is within two millimeters of the real value, considering the move of the leg over the stair in safe distance of twenty millimeters does not cause significant problems in stair climbing. A significantly higher value (+10mm) than the real size, could cause a problem of leg not being able to traverse the stair, and a significantly lower value (-10mm) could cause a problem in stability of the robot.

The depth of the stair is the most problematic value, as the total number of stairs is eleven, even the tiniest deviation can cause a fault by the end of the stair climbing, when the difference will cumulatively grow and result in a mistake in the foot placement. The problem with a too high value of the stair depth will eventually cause a push of the front legs against the following stairs, resulting in a position of rear legs out of the stair and a fall of the robot. So, the value estimated as the depth of the stair should be as close to real value as possible.

During the experimental evaluation, the most of the robot falls were caused by the rear legs running out of space on the stair. These problems appeared on the second half of the stairs, mainly at the end, where the cumulative error becomes to high. To demonstrate the possibilities of the robot, it was placed in front of the stairs and set to independently climb

the stairs; results are shown in Table 7.5.

Table 7.5: Success of stair climbing algorithm from various positions

No.	d_r	h	d	N	**Success**	**Time**
1	825	41	119	11	Yes	1:33
2	805	42	118	11	Yes	1:31
3	920	40	117	11	Yes	1:41
4	690	42	119	11	Yes	1:22
5	885	40	119	11	Yes	1:36
6	675	42	119	11	Yes	1:21
7	955	35	118	11	No	-
8	935	28	118	11	No	-
9	805	41	117	11	Yes	1:30
10	625	39	118	11	Yes	1:19

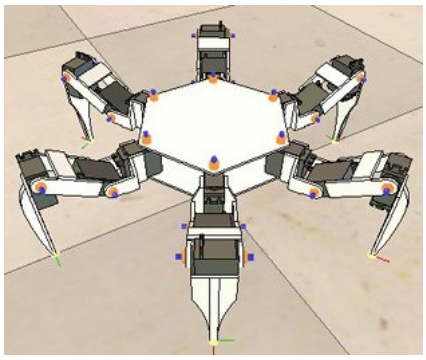
The falls in attempt 7 and 8 were caused by great difference between the estimated parameters and the real values. The errors did not appear immediately, since the robot was able to traverse eight steps until it fell down during the attempt number 7 and four steps during the attempt number eight. The quality of the image gets worse with increasing distance of the robot from the stairs, so, the position for taking a good quality image is from 470 mm to about 900 mm.

Chapter 8

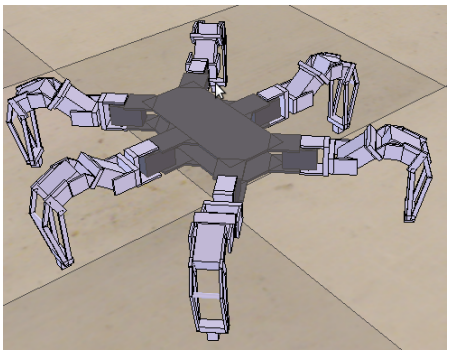
Simulation environment

The robot is not always available for testing the algorithm or the algorithm uses untried moves of particular actuator. In such a case, it can be helpful to use a simulation environment to perform initial evaluation of feasibility of the proposed ideas. In this thesis, we consider the open-source Virtual Robot Experimentation Platform (V-REP) [16] for such a testing.. The simulator is versatile, and objects created within the simulator can be controlled individually through embedded scripts, a ROS node, a remote API or a custom solution. Each controller can be written in C/C++, Python, Java, Lua or Matlab.

The simulator is able to generate additional information, such as collision detection, minimum distance calculation, data recording and visualization, which makes it very useful in work with model of the robot.



(a) predefined model [17]



(b) PhantomX model

Figure 8.0.1: Models in V-REP

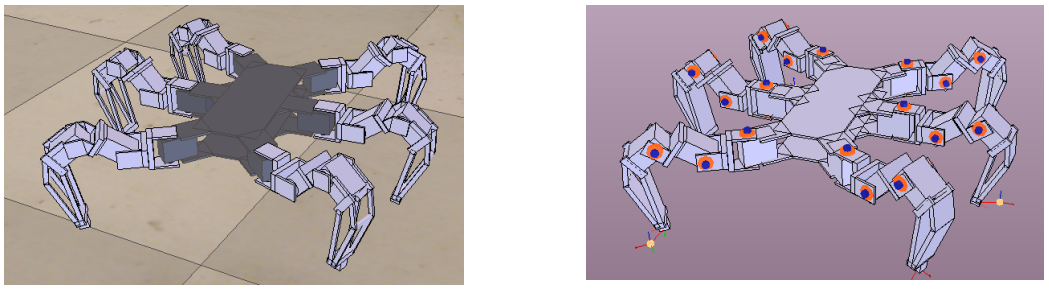
A model of the used robot was created using the tutorial for six-legged robot listed on the site of the producer [17], company Coppelia Robotics. The tutorials; however, uses a different six-legged robot as an example shown in Figure 8.0.1a, but the principle of the structure of the robot remains the same for our real platform. Therefore, the provided model has been adopted and a detailed model of the PhantomX has been created, see Figure 8.0.1b.

8.1 Model of the robot

The created model consists of two layers, which differ from each other during the simulation. The first layer is invisible to viewer during the simulation, because it does not precisely look like the real robot, as it is created only of pure shapes (cube, block, sphere, etc.). The basic shapes are easy to generate and calculate with; they are used for dynamics calculations of the simulation of the motion. Each shape is connected to a set of non-basic shapes in the second layer; so, the non-basic shapes are represented in the dynamics calculations, because they share the position and rotation in the simulation. The representation of the robot in these basic shapes is presented in Figure 8.1.1b, where each orange cylinder joint represents the rotary part of one actuator.

The second layer represents the visual appearance of the real robot. The position and rotation of each pure shape in the simulation is transmitted to the set of non-basic shapes; so, they are moved in the same way. A visualization of the second layer is depicted in Figure 8.1.1a.

The model consists of body of the robot and six identical legs that had to be modified to match the functionality of the real robot. The modification was done in settings of joints, as the left and right joints has to behave differently. The dimension of each part of the model was measured on the real robot and edited in the simulation environment to create authentic representation..



(a) visible layer

(b) invisible layer

Figure 8.1.1: Model of the robot

8.2 Control of the model

The robot in the simulation is controlled by Lua script, where each actuator is represented as an object and a new port is created in V-REP to receive the messages transmitted into the simulator. The messages are managed by the remote API functions [18] created by Coppelia Robotics for communication with the simulator.

References to all used objects and number of the created port are handed over to the program that controls the real robot, which has to be modified to be able to communicate with the simulator. The program is called within the script with all the objects and port number as parameters. The first thing that has to be done is a creation of the communication

bridge between the program and the simulator to allow testing the same program using simulated environment and real robot. It is done using the predefined function of the remote API and the number of port.

During the run of the simulation, the only necessary information that the simulator has to receive is the requested rotation of each actuator. As this information is sent to the real actuators, the only change that has to be done is a conversion of the value, which is sent to actuators, to a value in degrees. As it is required to rotate all the actuators at the same time, the communication between the program and the simulator is paused for time necessary to collect all the values of rotation and then sent together in one packet. After the program finishes its run, the communication bridge has to be shut down using a predefined API function.

There are still differences in behavior of the robot between the simulation and the real world, mainly considering the friction of the robot, as the motion of the legged robots is done using a tiny space on a foothold. Having a legged robot with permanent and guaranteed friction is also problem of real legged robots, as they are usually lightweight to consume less energy while moving and thus it can slip more easily than a heavier robot..

Chapter 9

Climbing down the stairs

The goal of this thesis is to design a motion gait for the hexapod walking robot to traverse stairs up. Although some ideas of the proposed approach can be considered for crawling down the stairs, there are significant differences between climbing up and down.

In the stair decline motion, the robot has to approach the stairs from the position near the top of the stairs, thus the stairs cannot be seen from a further distance. The robot would have to reach the end of the upper platform and change its body rotation, to be able to see the stairs with its camera; however, the principle used in the depth camera, does not allow to see close objects. So use of the depth camera is not straightforward..

Another way of identifying the parameters of the stairs, is using a leg of the robot itself, as the kinematics of the robot is precise. Each actuator is able to determine, whether a load is applied on the particular actuator. It does not only mean, that the actuator is pulling something, but it is also possible to identify, whether the actuator is pushing an object; in this case, it means pushing the body against the ground. The robot is able to distinguish between the position of the end of the leg in the air and standing on the ground. Using this feature, the surface in front of the robot can be searched for the edge of the upper platform and then the parameter of the height of the stair can be identified. After identifying the height of the stair, the next edge is found, which determines the depth of the stair parameter.

It is possible to use this search on every stair, but as it is assumed, all the stairs are identical, and the knowledge of the parameter shortens the time spent on the search, as it is performed only once. The number of the stairs parameter is not possible to estimate, prior traversing them without some exteroceptive sensor like the camera.

When all the stairs parameters are identified, the proposed approach for the climbing the stairs up can be eventually used as a base for climbing downstairs. The only difference is the negative value of the height of the stairs, because the depth parameter is supposed to remain the same, as the robot is still moving forward.

It is expected that an appropriate approach for downstairs climbing will also be an algorithm with three phases; however, the first phase needs to be modified entirely. The second phase should remain the same, except the fact that it is necessary to check, whether the robot reached the end of the stairs, after every traversed stair. The third phase also remains almost the same, as it brings the robot off the stairs, putting the rear feet on the ground, and

getting the robot ready for movement on planar surface.

These ideas need a further development and experimental validation, which is out of the scope of this thesis. The possibility of modifying the stair climbing algorithm for the purpose of climbing down the stairs can be set as a next goal for work with the robot.

Chapter 10

Conclusion

In this thesis, the problem motion control for a hexapod walking robot climbing upstairs has been studied. First, limits of such motion has been identified and a new motion gait has been proposed. Then, an autonomous detection of stairs parameters has been proposed and all the proposed ideas have been implemented and verified using real robotic platform.These limitations were found among the work with the robot and considered to be maximal for motion of the robot keeping the horizontal position of the body.

Based on the experimental evaluation, the robot could traverse even higher stairs, if it is allowed to change the rotation in the pitch angle of the body, which leads to a different approach to the movement of the robot. Regarding the rotation change, the robot increases limitations of the stair parameters, but at the same time loses some of its stability, as the footholds of the robotic platform PhantomX Mark II does not have enough friction, when the foot lands on the ground under a greater angle. A different approach is needed to allow the rotation of the body.

The proposed algorithm estimating the stairs parameters from a depth image seems to be a reliable source of information. It is able to get the necessary parameters to enable the robot to traverse the stairs. Beyond the assumption given for this task, it is not able to deal with very high noise around the stairs, e.g., people or objects standing next to the stairs, or with a camera facing the stairs under a wider angle. The problem of the camera is also a high vulnerability to heat sources, such as sun shining directly on the stairs occluding the pattern radiated by the camera, which results in a fault image. These problems are beyond the scope of this work and these topic can be further studied to enable a wider range of the traversing rough terrains by the robot.

The problem of the friction of the robot that should further studied. A light-weight robot loses a force to push the feet against the ground creating enough friction to prevent the feet from sweeping, which results in errors in robots movement, such as unexpected change in yaw rotation of the robot. This problem was slightly fixed by adding an additional battery pack on the robot and thus increases its weight.

The used simulation environment for the studied problem has been useful; however, only partially. It is not as precise as needed, because the performance of the model does not truly copy the behavior of the robot in real world. It can be used to validate smaller

movements of the robot, but the simulator is not as close to the reality as needed. The accuracy of the movement plays an important role in the stair climbing, and therefore, practical verification in real experiments is necessary even though it is more time consuming.

Regarding the future work with the robot, a creation of an algorithm based on a different technique, such as a machine learning, should be done, as there are still possibilities the robot is capable of. The results demonstrated that the approach used in this thesis can be improved to enhance robustness.

Bibliography

[1] K. Khoshelham, S. O. Elberink, Accuracy and resolution of kinect depth data for indoor mapping applications, in: MDPI, 2012.

[2] S. Oswald, A. Gorog, A. Hornung, M. Bennewitz, Autonomous climbing of spiral staircases with humanoids, in: Intelligent Robots and Systems, 2011.

[3] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, S. Schaal, Learning, planning, and control for quadruped locomotion over challenging terrain, in: The International Journal of Robotics Research, 2010.

[4] E. Z. Moore, M. Buehler, Stable stair climbing in a simple hexapod robot, Tech. rep., Ambulatory Robotic Lab, Centre for Intelligent Machines, McGill University (2001).

[5] MorpHex, (Cited on 31.3.2014).
URL http://zentasrobots.com/2014/03/17/morphex-mkii/

[6] X. Ding, Z. Wang, A. Rovetta, J. M. Zhu, Climbing and Walking robots, InTech, 2010, chapter 18 - Locomotion analysis of hexapod robots.

[7] ASUS, ASUS Xtion PRO, (Cited on 17.03.2014).
URL http://www.asus.com/Multimedia/Xtion_PRO_LIVE/

[8] Trossen Robotics, PhantomX Mark II platform, (Cited on 14.03.2014).
URL http://www.trossenrobotics.com/phantomx-ax-hexapod.aspx

[9] ROBOTIS, Dynamixel AX-18A Features, (Cited on 14.03.2014).
URL http://www.robotis.com/xe/dynamixel_en

[10] S. S. Roy, D. K. Pratihar, Modeling and analysis of six-legged rrobot; Analytical and soft computing-based methods, Lambert Academic Publishing, 2012.

[11] Vanadium Labs, Arbotix Commander, (Cited on 14.03.2014).
URL http://www.vanadiumlabs.com/commander.html

[12] Vanadium Labs, Arbotix Board, (Cited on 14.3.2014).
URL http://www.vanadiumlabs.com/arbotix.html

[13] OpenNI, (Cited on 30.4.2014).
URL `http://www.openni.org/`

[14] Y. Morvan, Acquisition, compression and rendering of depth and texture for multi-view video, Ph.D. thesis, Eindhoven University of Technology, The Netherlands, section 2.2. - Pinhole Camera Model (2009).

[15] Computer Vision Group, Munich Technical University, Intrinsic parameters, (Cited on 15.5.2014).
URL `http://vision.in.tum.de/data/datasets/rgbd-dataset/file_formats`

[16] Coppelia Robotics, V-REP, (Cited on 15.5.2014).
URL `http://www.coppeliarobotics.com/`

[17] Coppelia Robotics, Predefined Hexapod, (Cited on 15.5.2014).
URL `http://www.coppeliarobotics.com/helpFiles/en/hexapodTutorial.htm`

[18] Coppelia Robotics, Remote API, (Cited on 15.5.2014).
URL `http://www.coppeliarobotics.com/helpFiles/en/remoteApiOverview.htm`

Appendix A

CD Content

The enclosed CD contains the source code for implementation of the motion of the robot for stair climbing designed for PhantomX platform, the text of this thesis in a PDF format and source code of the thesis.

Table A.1: Directory structure

Directory	Note
/robot	The source files for the motion of the robot
/source	The source file of the thesis
/thesis	The thesis in PDF format