Master Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Terrain Modeling and Motion Planning for Hexapod Walking Robot Control

Diar Masri

Supervisor: doc. Ing. Jan Faigl, Ph.D.
May 2016

Acknowledgements

I would like to thank my thesis supervisor, doc. Ing. Jan Faigl, Ph.D., for his patience and advice while working on this thesis.

Declaration

Abstract

A planning technique for six-legged walking robot is presented in this thesis along with a terrain modelling approach that allows to exploit possible supporting footholds of the surface. The proposed approach for merging measurements provided by a depth camera sensor allows to reconstruct the robot surrounding. Based on the measurements of the environment, a grid based elevation map is constructed to create a suitable world model supporting motion planning for the six-legged walking robot. Such a model is adjusted to precisely represent a problematic passage in front of the robot by that provides only a limited number of possible locations for the robot footholds. Furthermore, an approach for an efficient utilization of the suitable positions of the terrain is proposed based on an evaluation function. Practical simplifications of the robot motion representation are proposed to efficiently speed up the planning of the robot forward motion over the terrain. Moreover, the proposed approach was successfully verified in real experiments.

Keywords: six-legged walking robot, point cloud, terrain modelling, motion planning

Supervisor: doc. Ing. Jan Faigl, Ph.D.

Abstrakt

Obsahem této práce je plánovací technika pro šestinohý kráčející robot spolu s postupem pro modelování terénu, který využívá možná místa v terénu pro podporu robotu. Navržený postup pro sloučení měření získaných z hloubkové kamery poskytuje přesnou rekonstrukci okolí robotu. Na základě kompaktních měření prostředí je sestaven vhodný model reprezentace prostředí pro plánování. Model je tvořen mřížkovou elevační mapou, která podporuje plánování pohybu pro šestinohý kráčející robot. Vytvořený model prostředí je dále upraven pro věrohodnější reprezentaci problematických částí ležících před robotem, které poskytují pouze omezené množství míst pro došlap nohy robotu. Dále je navržen postup pro efektivní využívání vhodných pozic terénu na základě hodnotící funkce. Navrženy jsou také praktická zjednodušení pro reprezentaci pohybu robotu, která efektivně zrychlují plánování pohybu vpřed. Navržený postup byl úspěšně ověřen reálnými experimenty.

Klíčová slova: šestinohý kráčející robot, point cloud, modelování prostředí, plánování pohybu

Překlad názvu: Modelování terénu a řízení šestinohého kráčejícího robotu v prostředí s úzkými místy došlapu

Contents

Figures

Tables

Introduction

The utilization of unmanned ground vehicles for the purpose of environment search in unstructured environment is an increasingly studied topic [1]. Such an approach allows people to explore dangerous or restricted areas [2] where wheeled or tracked vehicles are generally applied for the purpose of entering such locations [3]. In addition to the wheeled and tracked ground robots, walking robots provide a great flexibility to traverse rough and unstructured terrain (see Figure 1), such as regular stairs, debris, or even rocks. Such obstacles often provide only limited number of possible foothold positions therefore a specific motion control is required to encounter such obstacles. In comparison with biped or four-legged robots, the six-legged walking robots offer larger variety of walking styles with enhanced static stability, which is at the cost of a more complex control arising from the higher number of controlled actuators. Therefore, various motion planning techniques are studied [4].

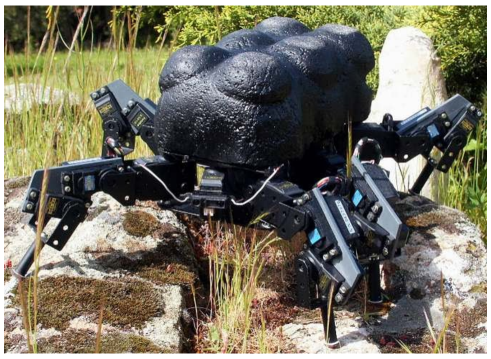


Figure 1: Six-legged walking robot

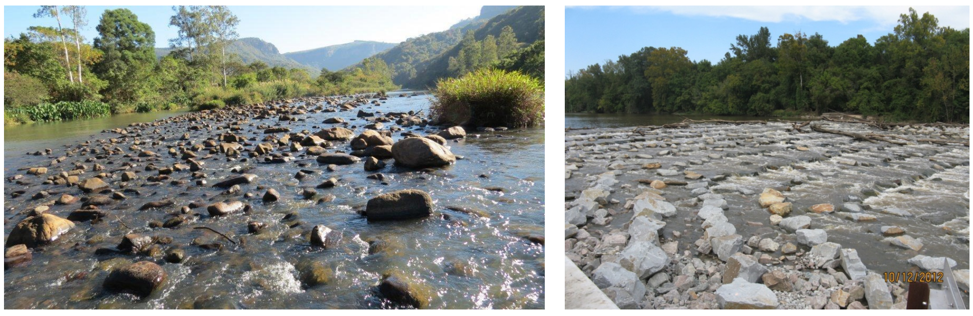
Two approaches can be applied regarding the robot motion planning and related terrain modelling: reactive and deliberative. The former approach uses proprioceptive sensors to enable the robot to quickly adapt to its surroundings and perform fast and stable short distance movements [5]. Due to the applied principle, the map of the robot surrounding is not considered. Therefore, it is unable to perceive feasible paths, even if they exist, due to the lack of information about larger surroundings.

The deliberative approach allows the robot to perceive larger areas by exteroceptive sensors, such as 3-D light detection and ranging (LIDAR) units, stereo camera or recently also RGB-D camera, which provides not only color image but also depth measurements. This approach requires a complex data processing of the provided sensor measurements to perform a particular sequence of motion actions for traversing the terrain [6], [7].

A reconstruction of the robot surrounding is necessary to allow selection of the most suitable path to safely traverse a rough terrain. Methods based on Simultaneous Localization and Mapping (SLAM) [8] [9] and Parallel Tracking and Mapping (PTAM) [10] techniques are one of the most applied approaches for the purpose of the position estimation and environment reconstruction in mobile robotics [11]. However, only relatively small amount of work has been done considering the mapping and localization with six-legged walking robots [12], [13], [14], [15].

Regarding the terrain modelling and motion planning for a six-legged walking robot addressed in this thesis, we are motivated by a problem of passing challenging terrains that can be found in nature, see Figure 2. The problem is to traverse a passage provided only with limited foothold locations. In such a situation, it is necessary to detect the feasible foothold locations and plan its motion to traverse the passage safely without falling down.

In this thesis, we propose to utilize RGB-D camera to reconstruct an environment in front of the robot and plan a safe path to crawl such a type of terrains. Moreover, we consider limited computational resources and we aim to develop on-line planning technique that can be deployed on-board of the utilized six-legged walking robot.



(a): **(b):**

Figure 2: Nature scenarios

Chapter 1
Problem Specification

The problem addressed in this thesis is to exploit the motion capabilities of the six-legged walking robot depicted in Figure 1.1. The greatest advantage of the six-legged walking robot is its flexibility considering the configuration of each part of the robot. Therefore, the robot is able to traverse an area which wheeled or track robot is not capable of.

The main objective of this thesis is to develop a motion control strategy that allows the robot to traverse a terrain with very narrow footholds, where it is necessary to precisely navigate the robot to such footholds, otherwise the robot can fall down. The following particular sub-problems can be identified to address the main objective of the thesis. The first sub-problem is to create a map of the robot surroundings. Then the second sub-problem is to develop a suitable traversability assessment together with an appropriate motion planning technique to provide a feasible motion plan that can be executed in real-time.

The problematic passage is located in front of the robot. An exteroceptive sensor is needed to enable information gathering about the surface properties that forms the passage. The RGB-D camera ASUS Xtion PRO [16] is considered as a suitable sensor for the purpose of this task regarding its measurement precision, dimensions, weight, and cost.

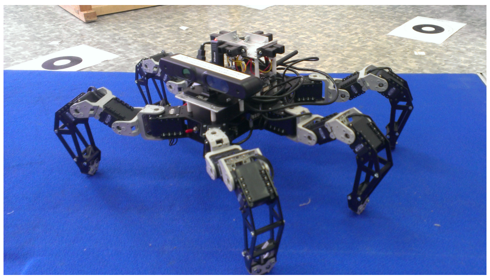


Figure 1.1: Position of on-board computer and RGB-D camera mounted on the robot

The on-board computer and the RGB-D camera represent the only load the robot has to carry. Hence, it is suitable to concentrate the weight of the load near the center of the robot body and thus, preserving the robot center of mass close to the center of the robot body, as it can be seen in Figure 1.1. Therefore, this set-up is considered as the initial conditions of the proposed approach.

Finally, a practical deployment of the developed solution provided path to a real robot is also part of the thesis goals. An experimental set-up, as shown in Figure 1.2, is considered for practical verification of the proposed approach. The robot has to build the map on-line, since only close vicinity of the robot can be captured and thus, the map of the robot surroundings has to be updated as the robot moves forward through the given passage. These are the initial conditions and assumptions for the addressed problem, which can be stated as: Develop a solution that will allow the six-legged walking robot to traverse a passage with only few and narrow footholds.

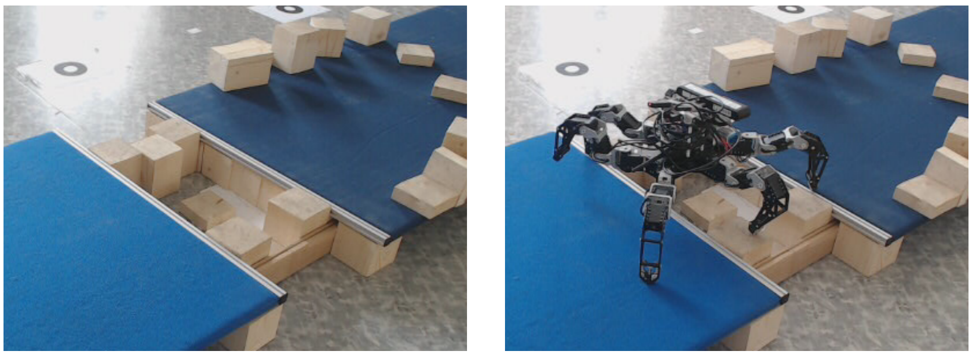

(a) : Passage detail **(b) :** Deployment demonstration

Figure 1.2: Laboratory testbed

1.1 Thesis outline

The overall approach is divided into consecutive sub-problems with specific goals that allows independent testing and evaluation of the particular parts. Regarding this, the text of the thesis is organized as follows. First, a short overview of the given hardware components to be employed for the verification of the proposed approach is presented in Chapter 2. A description of the used robot control is presented in Chapter 3. The proposed terrain mapping and traversability assessment are described in Chapter 4, which also includes practical issues arising from the physical limitation of the used sensor (Section 4.7). The proposed planning technique is in Chapter 5. Results of the experimental validation of the proposed solution are reported in Chapter 6. Furthermore, suggestions for possible improvements and future work are provided in Section 6.4.1. Finally, concluding remarks are dedicated to conclusion in Chapter 7.

Chapter 2

Hardware

The robot used for verification of the proposed approach functionality is built on the same platform used in [17]. Therefore, only a brief overview of the platform is given in this section along with additional changes applied to the motion control and available on-board computational resources. In addition, particular details about the robot sensor system are provided, because the utilized sensor has a direct impact to the proposed map building technique.

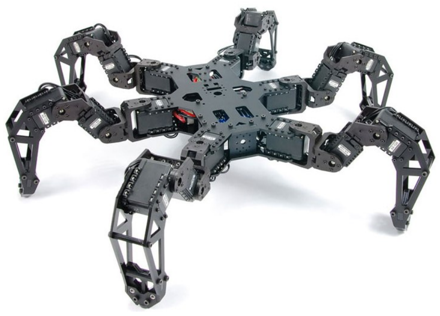


Figure 2.1: Robotic platform PhantomX Mark II

The used robot is based on the robotic platform PhantomX Mark II [18], which is a six-legged walking robot, see Figure 2.1, which is further described in Section 2.2. The robot consists of 18 servo drives that also directly represent the robot joints. Each joint is composed of a Dynamixel servo motor actuator, which is an intelligent actuator connected to a serial bus and capable of performing various control commands. A detailed description of the actuator is presented in Section 2.1. The ASUS Xtion PRO is considered as a suitable RGB-D camera for our needs and its properties are discussed in Section 2.3. As the computations are supposed to be done on-line, the computational board Odroid U3 is mounted on the robot and its features are outlined in Section 2.4.

2.1 Dynamixel AX-12A actuator

The Dynamixel AX-12A are the used intelligent servo motors [19] that control the motion of the robot. A picture of the servo is shown in Figure 2.2. The servos do not act only as simple actuators, but they are also able to work as sensors providing feedback information about the actual motion and performance, e.g., providing estimation of the current torque, position error, etc. The actuator is controlled by a microcontroller unit (MCU) and communicates through 3-wire connection, which is also used as the power supply.

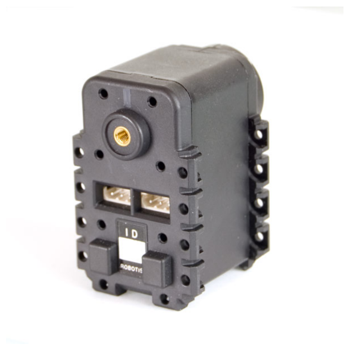


Figure 2.2: Dynamixel AX-12A actuator

2.1.1 Servo drive specification

A body of the actuator is made of the engineering plastic, which is a very resistant material. The plastic is also used for the system of gears propelled by the servomotor located inside the actuator along with the MCU ATMega8. The physical features of the Dynamixel AX-12A actuator are listed in Table 2.1.

Despite small dimensions of the actuator, it is capable of a high torque and high speed, at the cost of a high power consumption. 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 motion, the value can vary between 50 and 900 mA. This value mainly depends on the difference between the current position and the desired position of the servo motor, but also on the present load on the actuator. Notice, since the robot has 18 actuators, the peak load can be up to 16.2 A plus the power requirements of the on-board computational resources and power of the sensor system. Regarding the power supply and communication, each actuator is connected via three wires. One wire is used for power supply, one for ground and one for communication based on the transistor–transistor logic (TTL).

The actuators are able to set their goal position in a range between $-150°$ and $150°$ around the zero position as can be seen in Figure 2.3. The precision of the servo positioning depends on the resolution which corresponds to the third of a degree.

Table 2.1: Parameters of the Dynamixel AX-12A actuator

Model	**AX-12A**
Stall Torque	1.5 N.m^{-1}
Speed (RPM)	59
Nominal Operating Value	12 V
Stall Current Draw	1.5 A
Dimensions	$32 \times 50 \times 40$ mm
Weight	54.6 g
Resolution	0.29°
Operating Angle	300°
Gear Reduction	254:1
Position Sensor	Potentiometer
Com. Protocol	TTL
Com. Speed	1 Mbps

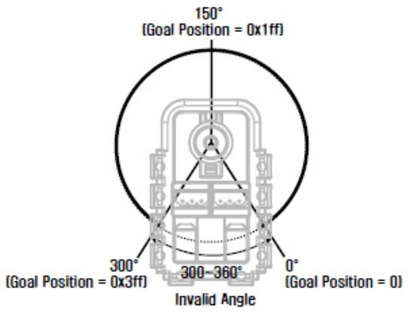


Figure 2.3: Operation range of actuator

2.1.2 Communication

The Dynamixel AX-12A is an intelligent actuator controlled by the TTL using the half duplex universal asynchronous receiver and transmitter (UART) serial communication. The actuators are connected to a serial bus and each actuator can be addressed by its unique ID. The ID of each actuator can be modified; however, the utilized motion control commands refer to actuators according to the schema shown in Figure 2.4.

The communication supports two types of packets: 1) the instruction packet and 2) the status packet. Each packet is divided into two parts: the header part and the data part. The structure of the instruction packet is visualized in Figure 2.5, 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 the ID of the addressed actuator, the length of the whole packet, the instruction, parameters related to the particular instruction and the checksum. The instruction defines whether the parameters are for

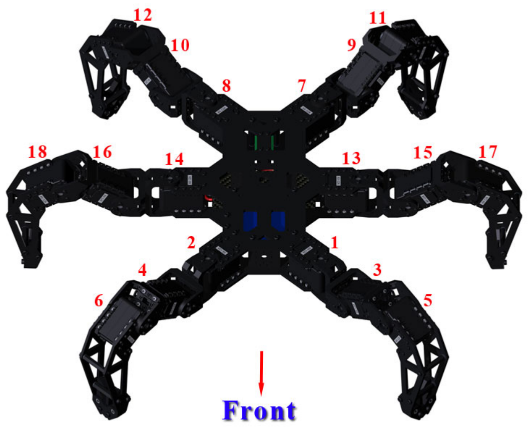


Figure 2.4: Positions of the platform's actuators

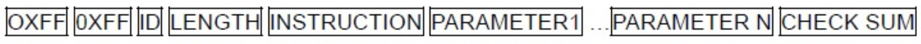


Figure 2.5: Instruction packet

read, write or synchronous write to more actuators at one time. Parameters represent optional data of the instruction packet, for example an address of the data to be read. The list of the most used addresses is presented in Table 2.2. Finally, the checksum is computed as an inverted sum of the bytes of the whole data part.

Table 2.2: 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 status packet is shown in Figure 2.6, where each box represents one byte. The structure serves as the response for the readings of the instruction packet and it has similar structure as the instruction packet. The header is formed by the first two bytes, which are used to mark the beginning of the packet. The data part consists of the ID of the previously addressed actuator,

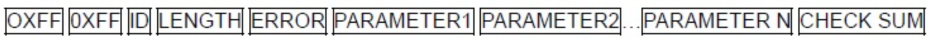


Figure 2.6: Status packet

the length of the whole packet, a possible error in the actuator, parameters requested for a particular instruction and the checksum. If an error appears, there is an information about it and its origin in the error byte. The MCU in the actuator can differ between errors that happened, e.g., overheat of the actuator, not enough input voltage, etc. All possible return error values are listed in Table 2.3.

Table 2.3: Error details

Bit	Name	Detail
7	0	No error
6	Instruction Error	Set to 1 if an undefined instruction is received.
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 received instruction is out of the defined range.
2	Overheat Error	Set to 1 if the internal temperature of the unit is above the allowed operating temperature.
1	Angle Limit Error	Set to 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.

The maximum number of actuators connected to a single serial bus is 254, as the possible ID is in the range 0–253, the ID number 254 is reserved for the broadcast communication when a packet is addressed to all connected actuators at once. The limitation on the number of actuators arises from the fact that the ID is represented as a single byte, and the number 255 is reserved for beginning of the packet.

The communication can be performed on different baud rates. The computation speed can be controlled by setting a dedicated address (Address4) of the MCU to a specific value from which the communication speed is determined according to

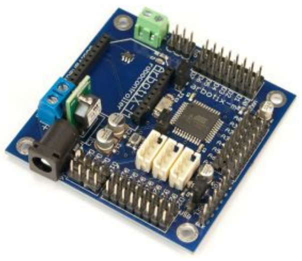
$$Speed = 2000000/(Address4 + 1), \tag{2.1}$$

which sets the baud rate value, where the maximum baud rate error of 3% is within the tolerance of the UART communication standard. The initial value of the address is set to 1, which defines the default communication speed as 1 Mb.s^{-1}.

The Dynamixel actuator does not work only as a simple actuator, but also acts as an intelligent sensor. When any of the parameters exceeds its limit, the actuator can move again, if the reason is temporary, such as exceeding the angle limit. The actuator moves again, when the next requested goal position, inside the limits, is received. There are however problems that can be solved only by a hardware restart of the actuator, e.g., exceeding the maximum load. Simultaneously to the reported error in the status packet, LED of the servo lights to inform a collision occurred on the actuator.

The communication between control unit and the robot can be done in several ways based on the particular option for the used control computer.

- Microcontroller unit - used in the original robot kit [18], where control of the servo drives was done by a controller [20] that gave instructions for Arbotix board [21] (MCU), presented in Figure 2.7a.



(a) : Arbotix Board

(b) : Xbee radio module

Figure 2.7: Communication routes between control unit and the robot

- Xbee communication - a wireless communication, where a control unit sends data to Xbee transmitter and Xbee receiver is mounted on the robot, the Xbee radio module is shown in Figure 2.7b. This type of communication does not directly support a half-duplex communication used for connecting servo drives to the serial bus.

- USB - a control unit sends a command through a USB cable, which is transferred to the TTL by the USB2DYNAMIXEL adapter shown in Figure 2.8.

In this thesis, the last option of the communication is utilized for a reliable communication between the servo drives and control computer.

Figure 2.8: USB2DYNAMIXEL adapter

2.2 Robotic platform PhantomX Mark II

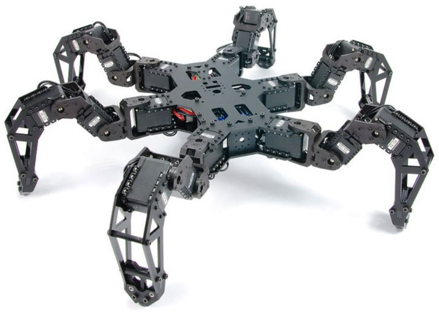


Figure 2.9: Robotic platform PhantomX Mark II

The robotic platform PhantomX Mark II [18] is a six-legged walking robot with the dimensions of the body $240 \times 120 \times 39$ mm and it is shown in Figure 2.9. Particular 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 (DOF), which allows it to place the foot in various positions. However, there are limits for the motion, which arise from the leg construction.

Each leg consists of three Dynamixel AX-12A 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, see Figure 2.10. The name for each actuator is from Latin name of the particular part of an insect body, which performs similar motion as the corresponding part of the robot leg.

2.2.1 Practical issues

We made following modification of the original PhantomX platform to make it usable for the experimenting with the proposed approach. The original version was controlled by the Arbotix board [21] and remote controller [20] that gave instructions to the Arbotix board (MCU). The actuators were connected directly to MCU, which has no usage in this task, as the main feature of the

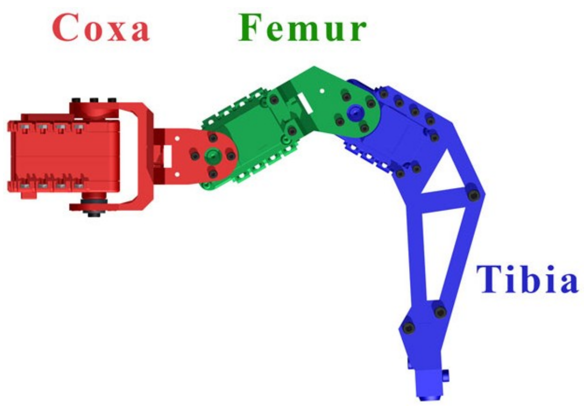


Figure 2.10: Names of each part of leg

proposed solution should be autonomy. Therefore, these components have been replaced by a small on-board computer Odroid U3 further described in Section 2.4.

The power supply used for powering the robot has to serve both for powering the actuators of the robot and to power the on-board computer mounted on top of the robot. When testing individual parts of the approach, powering from electricity network is used, which is more comfortable considering exchanging of parts. However, a LiPol battery pack with 4 Ah has been used to test the entire proposed autonomous navigation and to eliminate any external forces applied on the robot, such as tensioned cable of the power supply. It has been observed in early experimental evaluation that such additional forces can change the heading or position of the robot significantly.

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.

2.3 ASUS Xtion PRO

The ASUS Xtion PRO [16] is a RGB-D camera considered for the perception of the surface in front of the robot. It is based on the Primesense sensor which is also utilized in the probably more popular (but significantly larger and heavier) Microsoft Kinect. It is a system that provides multiple 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 serves as power supply. The depth images provided by the camera are very useful considering the addressed task, because it is able of transforming the depth image into a point cloud.

The depth sensing is based on infra-red (IR) projector and IR sensor. The camera consists of one color image sensor and one IR sensor accompanied by

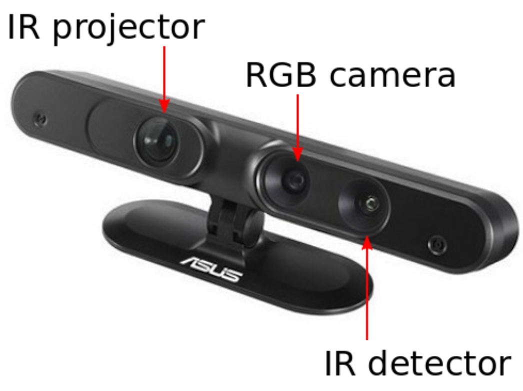
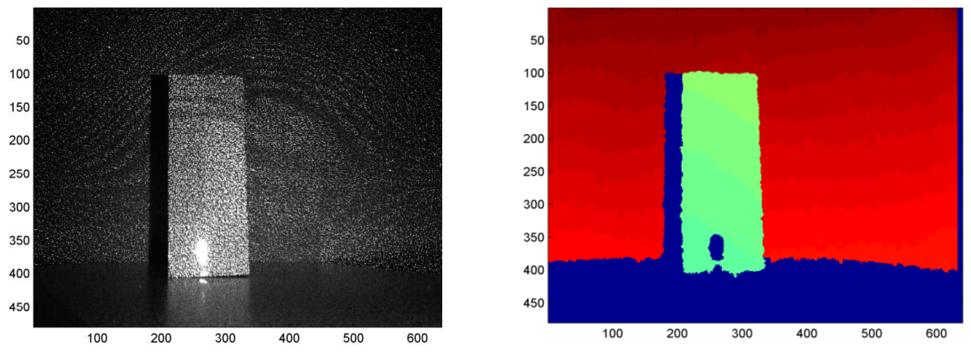


Figure 2.11: ASUS Xtion PRO

IR projector, see Figure 2.11. 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 IR projector projects a pattern of IR dots, see Figure 2.12a, which falls on all objects in the range of the camera, the dots are detected using a conventional CMOS image sensor. The pattern changes its size and position based on the distance of the objects from the source (the IR projector). The depth sensor generates depth value for every pixel, as it is shown in Figure 2.12b. 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.



(a) : IR image of the pattern **(b) :** Depth image

Figure 2.12: IR pattern projection into the depth scene where far objects are in red while close objects are in blue.

The application programming interface (API) for ASUS Xtion PRO is provided by Open Natural Interaction (OpenNI) [22] 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, e.g., to read the image 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 0.5 metres. The reason is that the projected pattern is so bright that the camera is not able to recognize it. The depth sensor is not able to evaluate the data; so, they are assigned to zero distance.

In the addressed problem, such a disadvantage of the measuring device is compensated by building a map of the terrain in front of the robot and based on several consecutive images of the scene for different positions of the robot. Therefore, such an algorithm is capable of coping with the disadvantage of the measuring approach used by the camera.

2.4 Hardkernel Odroid U3

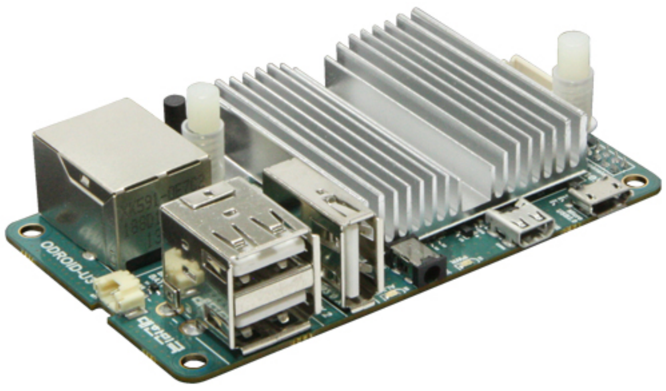


Figure 2.13: Hardkernel Odroid U3

The Hardkernel Odroid U3 platform [23], shown in Figure 2.13, is mounted on the robot to enable on-board computations. The platform joins the input from the camera with planning of the motion resulting in appropriate output sent to the Dynamixel actuators.

Despite its small dimensions, the computer consists of 1.7GHz quad-core processor accompanied by 2GB RAM that provides enough computational resources for the solution proposed in this thesis. Detailed specification of the Odroid U3 is listed in Table 2.4. As the storage capability of the computer is only optional, the computer is equipped with additional 8GB MicroSD card that provides enough capacity for operating system and additional space for storing data created through the proposed approach, i.e., point clouds, maps and data about the proposed motion.

Regarding the operating system installed on the platform, the Linux based Ubuntu 14.04. LTS operating system designed for Advanced RISC Machine (ARM) architecture has been considered due to the availability and support

Table 2.4: Specification of the Hardkernel Odroid U3

Hardkernel	**Odroid U3**
Processor	Samsung Exynos4412 Prime Cortex-A9
	Quad Core 1.7Ghz with 1MB L2 cache
Memory	2048MB(2GB) LP-DDR2 880Mega data rate
LAN	10/100Mbps Ethernet with RJ-45 Jack
Storage	MicroSD Card Slot
	eMMC module socket
Power	5V 2A Power
PCB Size	83 x 48 mm
Weight	48g including the heat sink

of Robot Operating System (ROS) [24] and simple connectivity with external devices.

The Indigo version of ROS was considered as it was originally designed to work under the Ubuntu 14.04. without any additional adjustment. The ROS Indigo is a framework used for inter process communication and also as the communication layer providing data streams from the ASUS Xtion PRO, using the OpenNI development kit.

The powering of the computational board is shared with powering of the actuator system through a Li-Pol battery pack with the capacity of 4 Ah. The powering solution is a simple construction with easy maintenance; however, in the case of exchanging the battery, or changing the communication route to the actuators, the platform has to be rebooted. The distance covered by the robot during a single test of the approach is not large enough to cause an immediate exchange of batteries in the particular experiment, therefore, the disadvantage caused by such a powering solution is not considerable.

Additional advantage of having all necessary resources and computational power on board of the robot is elimination of any external force that causes undesirable change of the robot orientation or position, resulting in an error in the proposed precise foothold placement. Furthermore, it enhances the effort in creation of the fully autonomous robot able to crawl over complex obstacles.

Chapter 3

Robot Control

The main requirement for the robot motion is the stability of the robot; therefore, the motion is based on pentapod walking gait [25]. In the pentapod gait, at least five legs remain in contact with the ground and one leg at a time performs a step. However, the robot legs do not perform a constant motion as in a regular gait, but the position of the next foothold is planned, to utilize the terrain more efficiently and to allow the robot traverse a passage with narrow footholds. The body moves simultaneously with the leg that performs a step; moreover, a synchronised motion of each actuator is applied; thus, a speed of the particular actuator is changing during the motion. Such approach reduces the possibility of collision of the robot with itself and enhances the a smooth motion.

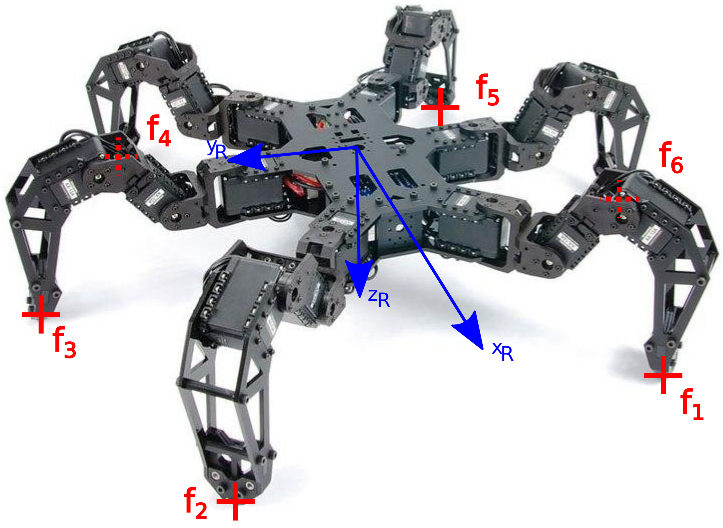
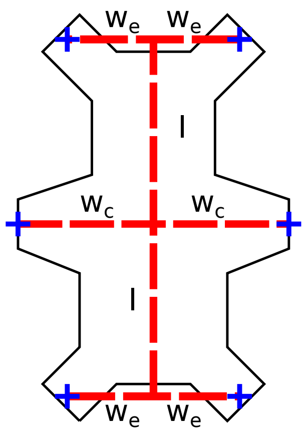


Figure 3.1: Coordinate system of the robot R with the particular endpoints denoted as $f_i, i \in (1, \ldots, 6)$

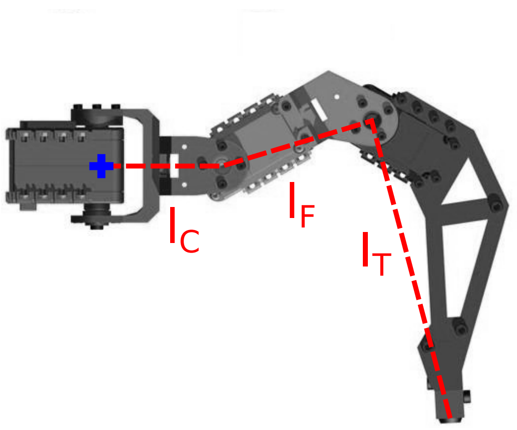
The construction of the robot consists of 18 actuators representing 6 legs, each with 3 actuators. The considered gait (with one moving leg and five supporting) divides the 18 parameters representing the actuators into two groups: 3 parameters represent the leg performing a step and 15 actuators are performing the forward body motion. The configuration of each leg is defined by three parameters either in joint coordinate frame $s(C, F, T)$, suitable for control, or in Cartesian coordinate frame $f(x_R, y_R, z_R)$, suitable for planning. The conversion from the Cartesian coordinate frame to the joint coordinate frame is done by the inverse kinematic task (IKT). The Cartesian coordinate system of the robot R is shown in Figure 3.1. A position (in the R coordinate frame) representing the endpoint of each particular leg is defined as a triplet

$$f_i = x_i^R, y_i^R, z_i^R, \tag{3.1}$$

where i denotes the particular leg, i.e.,$i \in \{1, \dots, 6\}$ with the origin at the center of the robot body.



(a) : Geometrical model of the body



(b) : Geometrical model of the leg

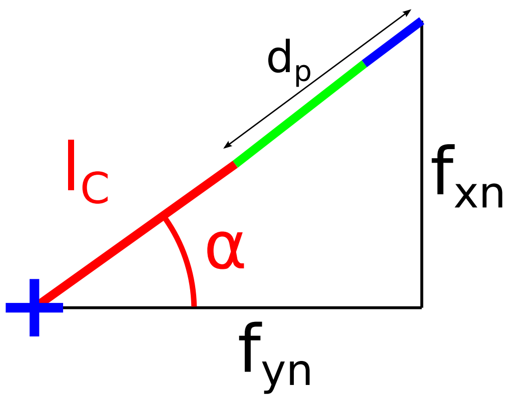
Figure 3.2: Geometrical model of the robot, where corresponding parts are in blue.

Even though, each leg is attached to the robot in a different way and cannot be treated identically regarding the IKT, the main idea of computing joint values is the some for all legs. Therefore, the IKT is described on one leg, other legs are treated analogically using a coordinate transformation of the particular leg.

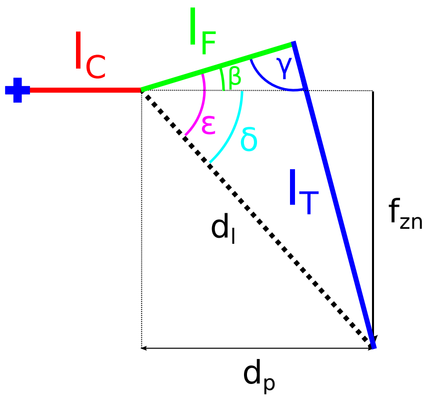
The angle α, presented in Figure 3.3a, defines the orientation of a coxa servo drive and is calculated as,

$$\alpha = \tan^{-1}\left(\frac{f_{xn}}{f_{yn}}\right). \tag{3.2}$$

Finding α, the transformation becomes a 2D task that can be approached using a cosine formula, see Figure 3.3b. The angles β and γ, which define the orientation of femur and tibia servo drives, are calculated using auxiliary parameters



(a) : Top view of the leg



(b) : Side view of the leg

Figure 3.3: Geometry of the leg for the IKT

$$d_p = \sqrt{f_x^2 + f_y^2} - l_C \tag{3.3}$$

$$d_l = \sqrt{d_p^2 + f_z^2} \tag{3.4}$$

$$\delta = -tan^{-1}\left(\frac{f_z}{d_l}\right), \tag{3.5}$$

in the cosine formula

$$l_T^2 = d_l^2 + l_F^2 - 2d_l l_F cos(\epsilon). \tag{3.6}$$

Expressing ϵ as

$$\epsilon = \cos^{-1}\left(\frac{d_l^2 + l_F^2 - l_T^2}{2d_l l_F}\right) \tag{3.7}$$

the angle β is

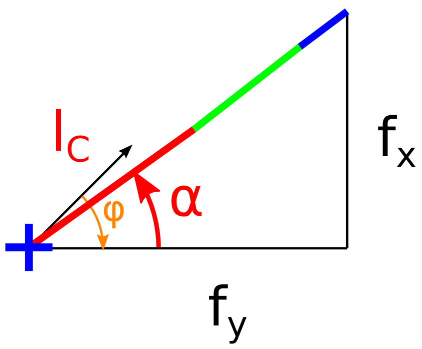
$$\beta = \delta + \epsilon \tag{3.8}$$

and analogically, the angle γ is calculated using the cosine formula

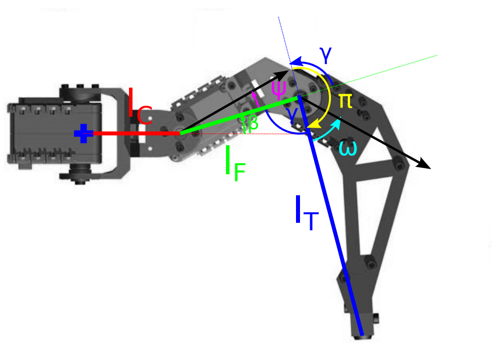
$$d_l^2 = l_F^2 + l_T^2 - 2l_F l_T cos(\gamma) \tag{3.9}$$

$$\gamma = cos^{-1}(\frac{l_F^2 + l_T^2 - d_l^2}{2l_F l_T}). \tag{3.10}$$

The geometrical model of the leg, presented in Figure 3.2b, is a simplification of the real leg as the connections between the adjacent actuators are not straight segments. Therefore, a bias has to be applied to set the actuators to the particular requested value. The offset angles are depicted in Figure 3.4a and 3.4b where they are denoted as the angles ϕ, ψ, ω, π.



(a) : Top view



(b) : Side view

Figure 3.4: Direction of particular angles, where the default orientation of particular actuator is represented by black arrow.

Table 3.1: Angle adjustment

f_i	o_C	a_C	o_F	a_F	o_T	a_T
f_1	ϕ	$-\alpha$	$-\psi$	$-\beta$	$-\omega+\pi$	$-\gamma$
f_2	$-\phi$	α	ψ	β	$\omega-\pi$	γ
f_3	0	α	ψ	β	$\omega-\pi$	γ
f_4	ϕ	$-\alpha$	ψ	β	$\omega-\pi$	γ
f_5	$-\phi$	α	$-\psi$	$-\beta$	$-\omega+\pi$	$-\gamma$
f_6	0	$-\alpha$	$-\psi$	$-\beta$	$-\omega+\pi$	$-\gamma$

The angle of a particular actuator must be in the range $\langle -150°, 150° \rangle$ with the center of the range shown as the black arrow in Figure 3.4. The particular angles are therefore adjusted using the offset values ϕ, ψ, ω, π to correspond with the control of the actuator. However, the legs differ in their construction with respect to the robot body, therefore, the parameters of the adjustment have to be uniquely applied. Particular offset angles are reported in Table 3.1. The angle of the particular actuator is then calculated as

$$s_{iC} = 150 + o_{iC} + a_{iC} \tag{3.11}$$
$$s_{iF} = 150 + o_{iF} + a_{iF} \tag{3.12}$$
$$s_{iT} = 150 + o_{iT} + a_{iT}. \tag{3.13}$$

Neither of the actuators of the robot is able to utilize the full interval of its reachable positions, due to the leg construction. Therefore, it is necessary to eliminate any attempt to set the actuator to a position resulting in a collision with adjacent parts of the robot body, as it is indicated in Figure 3.5.

Each operation with the angle of the particular actuator is preceded by a check, whether the required angle lies within the reachable interval, specifically set for the each particular actuator. The intervals for particular actuators are

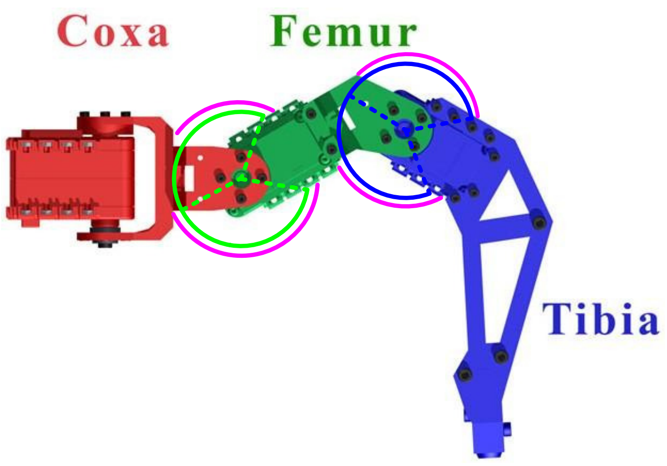


Figure 3.5: Construction limits of the femur and tibia actuator. The red,blue and green parts represent particular parts of the leg. All reachable positions of the particular actuator are represented in magenta.

listed in Table 3.2 and considered in the motion planning. Therefore, every leg configuration defined by the particular joint angles $s_i = (s_{iC}, s_{iF}, s_{iT})$ has to satisfy the conditions.

$$s_{iCmin} \leq s_{iC} \leq s_{iCmax} \tag{3.14}$$
$$s_{iFmin} \leq s_{iF} \leq s_{iFmax} \tag{3.15}$$
$$s_{iTmin} \leq s_{iT} \leq s_{iTmax} \tag{3.16}$$

Once the IKT is formulated, the leg performing a step has to be provided by coordinates $f = (x, y, z)$ of a new foothold. The candidates for the next foothold are determined according to the terrain in front of the robot.

Table 3.2: Used safe limits of the particular leg actuators

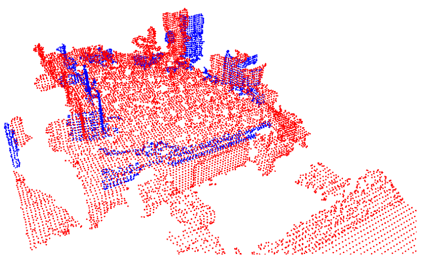
f_i	s_{Cmin}	s_{Cmax}	s_{Fmin}	s_{Fmax}	s_{Tmin}	s_{Tmax}
f_1	-80	42	-105	101	-55	101
f_2	-42	80	-102	103	-104	55
f_3	-80	80	-104	102	-104	55
f_4	-80	42	-104	101	-105	55
f_5	-42	80	-104	102	-55	99
f_6	-80	80	-104	102	-55	99

Chapter 4

Terrain Modelling

Terrain modelling together with localization of the robot in such a model plays a key role in the robot navigation task, mainly in the unstructured environments, where surroundings of the robot cannot be modelled by a simple plain surface. The RGB-D camera ASUS Xtion PRO has been selected as a suitable source of range measurements, because it represents an off-the-shelf low-cost sensor able to deliver sufficiently precise measurements for the addressed problem..

Once we have a precise position of the robot, it is relatively straightforward to build a terrain map from consequent measurements (3D scans provided by the RGB-D camera). We only need to translate particular scans according to the changed robot position $P(x, y)$ performed between the measurements and merge the scans into a single terrain map. However, we do not rely on external motion capture system, and therefore, we only have the collected scans and expected motion $\tilde{P}(x, y)$ provided by the motion planner, which does not truly correspond to the real robot motion due to friction of footholds, errors in position settings of the actuators, etc. Considering only the expected motion change $\tilde{P}$ provides a map with insufficiently precise alignment of the individual scans, which is demonstrated in Figure 4.1a.


(a) : Simple translation of point clouds where aligned cloud is in red

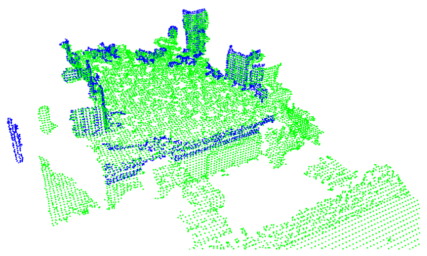

(b) : ICP alignment where aligned point cloud is in green

Figure 4.1: Alignment comparison between simple translation and ICP where target point cloud is in blue.

On the other hand, we can consider SLAM techniques to simultaneously built a map of the environment and use the map to provide estimation of the robot motion. Thus, a point cloud C_t of the new measurements provided by the sensor is aligned to the point cloud C_s that represents the world model (map), e.g., using the iterative closest point (ICP) [26] algorithm, which results in much better map as it is shown in Figure 4.1b. Therefore, map building technique based on the ICP algorithm is utilized in this thesis to create a model of the robot surroundings.

The algorithm works as follows, a point cloud C_t of the new measurements provided by the sensor is aligned to the point cloud C_s that represents the world model. The algorithm iteratively converges to a transformation Q that transforms the point cloud C_s to align with point cloud C_t by minimizing the mean squared error between them. Based on early experiments, the transformation Q is more precise, when normals are considered instead of points; therefore, a normal is estimated for each point of both point clouds. Once the transformation Q is estimated and the point cloud C_s is aligned to C_t; the map point cloud C_s is updated by merging the point cloud C_t into C_s. It can be assumed that points in a close vicinity to each other are identical, and therefore, a voxel[1] grid approach is applied to create a 3D voxel grid [27] and remove the duplicity of particular points. Such an update of C_s integrates sensor measurements that can contain noisy data; therefore, the moving least square (MLS) [28] method is applied on the 3D voxel grid to reduce noise in the created world model. Beside the terrain model, the transformation Q enables the robot to simultaneously localize itself with respect to the terrain and provides better position estimate than the expected motion $\tilde{P}$.

Even though, the world model represented by the point cloud is good for localization, it is difficult to plan a robot motion in a point cloud. In particular, we assume that the robot uses only the top surface of the terrain and obstacles for crawling, and therefore, it is possible to transform the point cloud into a grid based elevation map which represents the 2.5D space which is more suitable for the proposed on-line motion planning.

As it is considered that the terrain provides limited number of foothold locations, a method that indicates the most suitable candidates for foothold placement is proposed. The perceived rough terrain map is divided into two groups of accessible and non-accessible foothold locations to create the so-called foothold map. Then, an evaluation function is proposed to consider expected robustness of a particular foothold. Such as map is further considered in the motion planning.

The motion planner accesses the evaluation function, regarding the position (x, y) for the foothold placement, and the elevation map, also regarding the height of the particular position $\mathcal{M}_{elev}(x, y)$. Therefore, the motion planner is provided with a full triplet (x, y, z). The data flow of the terrain modelling and a relation of the particular terrain maps with the motion planner motion planner are depicted in Figure 4.2.

[1]Voxel is a point in a regular 3D space representing a predefined area in the space, it is analogous to a pixel in the 2D space.

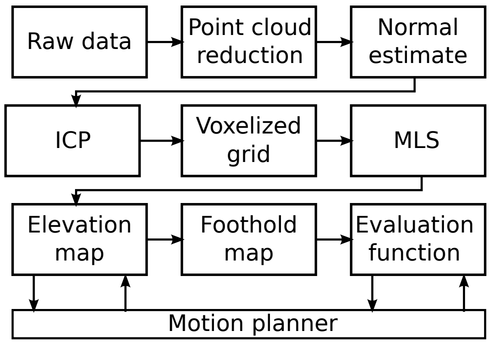
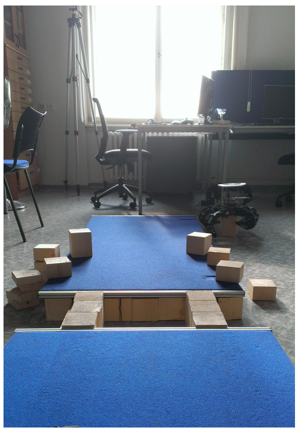
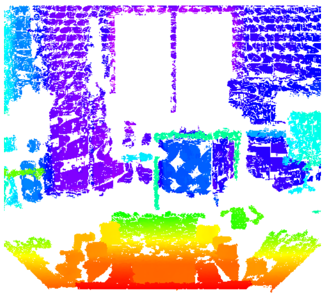


Figure 4.2: Data flow diagram of the terrain modelling and motion planner

4.1 Point cloud model



(a) : Real terrain in front of the robot



(b) : Raw point cloud generated by ASUS Xtion PRO

Figure 4.3: Terrain in front of the robot

The ASUS Xtion PRO camera is able to provide a depth image in the resolution of 640×480 pixels; furthermore, it is also able to convert it into a raw point cloud scan, which is subset of $\mathbb{R}^3$ that consists of 307200 points. An example of the raw point could directly provided from this RGB-D sensor is shown in Figure 4.3b. The credibility of the measurement can be compared with the real terrain in front of the robot presented in Figure 4.3a, which is capture by the RGB sensor of the utilized Asus Xtion Pro. The origin of coordinate system is aligned with the sensor device, as it is simple to compare the measured distances with the real scene. The coordinate system, is oriented as it is shown in Figure 4.4.

The pitch of the camera can be arbitrarily set, using the joint of the camera holder construction. The RGB-D camera is set in a manner that the z-axis is perpendicular to the ground. This form of the coordinate system setting makes the transformation between the coordinate system of the camera and the coordinate system of the robot a simple translation.

4.1.1 Point cloud reduction

The amount of data provided by the camera is enormous for the purpose of addressed modelling of the terrain in front of the robot. Thus, a raw point cloud is reduced to a lower number of points considering only points inside the box with the dimensions of $(2.7, 0.75, 0.75)$ metres in front of the robot. The raw points in this box are highlighted in Figure 4.4.

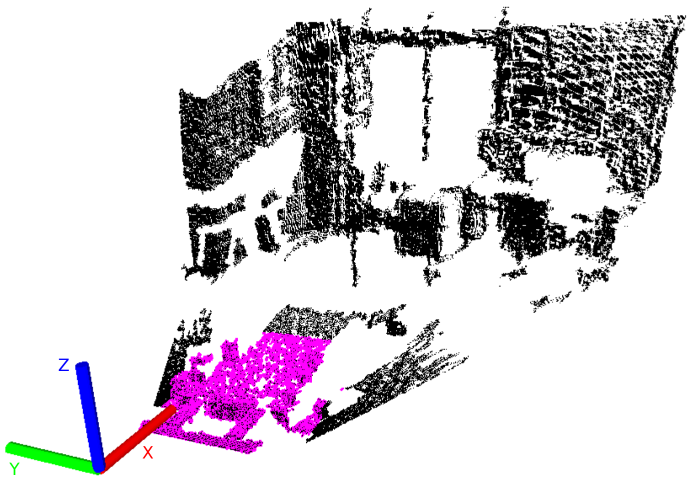


Figure 4.4: An example of the point cloud reduction where X-axis is in red, Y-axis is in green and Z-axis is in blue. The raw point cloud provided by the camera is represented in black and the reduced point cloud within a selected area is in magenta.

4.2 Iterative Closest Point

The ICP algorithm, particularly its implementation using the Point Cloud Library (PCL) [29], is used to iteratively refine a source scan C_s to match a target scan C_t in the l iterations. In each iteration $k \in \{1, \dots, l\}$, the algorithm estimates corresponding features between the point clouds C_s and C_t, estimates the rotation matrix $R_k(\phi, \psi, \omega)$ and the translation matrix $T_k(x, y, z)$ while minimizing the mean squared error e_k and transforms the point cloud C_s using the estimated transformation matrices R_k and T_k.

The algorithm matches point $p = (x_1, y_1, z_1), p \in C_s$ to point $q = (x_2, y_2, z_2), q \in C_t$ using the Euclidean distance as

$$q = \arg\min_{r_i \in C_t} \|p - r_i\|. \tag{4.1}$$

The point q with the minimal Euclidean distance to the point p is registered as tuple (p, q). In each iteration k, the algorithm minimizes the mean squared error e_k

$$e_k = \frac{1}{N} \sum_{i=1}^{N} \|q_{ik} - R_k p_i - T_k\|, \tag{4.2}$$

where N is the number of points in C_s.

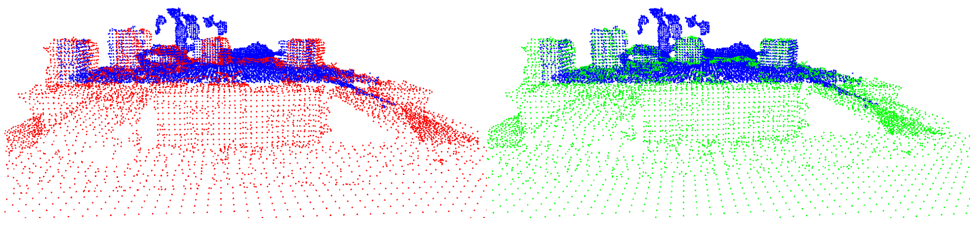
The ICP algorithm always converges monotonically to the nearest local minimum of the mean-square distance metric [26]. Once converged to a local minimum a transformation Q between the initial point cloud C_s and point cloud C_t is estimated based on the rotation matrices $R_{1,\ldots,l}$ and translation matrices $T_{1,\ldots,l}$.

The robot motion in unstructured terrain is considered, therefore, it is assumed that the terrain consists of various objects with unique surface, which provides more distinguishable features. Therefore, normals are estimated in each point of both point clouds. The normal n_p at the point p is estimated as

$$\vec{n}_p = \vec{v}_{1p} \times \vec{v}_{2p}, \tag{4.3}$$

where $\vec{v}_{1p}$ and $\vec{v}_{2p}$ represent vectors between the point p and its first and second nearest neighbour.

Using normals for estimation of the transformation Q results in a better performance of the ICP algorithm. The precision of the alignment using normals is shown in Figure 4.5b in comparison with the simple ICP presented in Figure 4.5a. Once the ICP algorithm converges, the point cloud C_s and C_t are merged and the current robot position is updated according to the found transformation Q.



(a) : Simple ICP with source point cloud in red and target point cloud in blue

(b) : Registration using normals where source point cloud is in green and target point cloud in blue

Figure 4.5: Alignment comparison between point and normal-based registrations method; the difference between the alignment is only in centimetres; however, such a distance is significant considering the size of the robot body.

4.2.1 Practical implementation

The initial estimate of the transformation Q, i.e. T_0, during the robot motion is provided by the expected motion $\tilde{P}$ if available. Therefore, the algorithm provides better results in the fixed number of 50 iterations, due to the lower mean squared error, and should not get stuck in a local minimum. The control of the robot motion shows a high precision with a proper traction; thus, the motion estimate $\tilde{P}$ should be close to the real robot motion performed since the last measurement.

Due to the considered performace, an implementation of the ICP algorithm considering normals at particular points of the point cloud is employed [30], along with the algorithm for the normal estimation [31] which is based on neighbourhood technique.

Even though the raw point cloud, received from the camera is reduced (see Section 4.1.1), the size of the both point clouds C_s and C_t is still too large to be processed by the ICP in a reasonable time. Therefore, the point clouds are further down-sampled, to improve performance of the proposed solution, which is made using the PCL.

4.3 Voxel grid

The voxel grid approach is applied to create an elevation map as 3D voxel grid. All points belonging to a particular voxel are approximated with their centroid. This approach is a bit more computationally demanding than approximation of the points in the voxel as the center of the voxel, but it represents the terrain measured by the points of the point cloud more accurately.

The voxel grid is then utilized for two purposes regarding the addressed problem of this thesis. The first purpose is to enable a better performance of the ICP algorithm, as the reduced number of points is used for the transformation estimate, in which a voxel grid with the dimensions of $(0.05, 0.05, 0.05)$ metres is considered. The second purpose is to remove the duplicated points after merging the point clouds C_s and C_t together. The voxel grid with dimensions of each grid bin $(0.01, 0.01, 0.01)$ metres is utilized to filter the updated the point cloud C_s.

Finally, the moving least square algorithm is employed to filter a possible noise in the voxel grid. The position description (x, y, z), together with the normal (n_x, n_y, n_z) is stored in each bin of the grid. The information about normals makes the planner able to perceive the supporting possibilities of the terrain more precisely.

4.4 Grid based elevation map

The elevation map describes altitude of each particular grid cell and thus, it provides a discrete representation of the spatial distribution of the terrain. Each grid cell represents a particular foothold location whose suitability to stably support robot position can be determined using nearby cells. The elevation map is created with the considered dimensions 2.7×1.0 meters, which is sufficient for covering the considered area in front of the robot. Each grid cell represents a squared area with the side 0.01 meter that corresponds to the dimensions of the voxel grid (Section 4.3) that is converted to the elevation map.

In a case of multiple hits in a particular bin of the elevation map, the maximum rule is applied considering the z coordinate, as the movement on top of the obstacles is considered. Regarding the normals of multiple hits a mean filter is further applied to smooth the map. An example of the created elevation map $\mathcal{M}_{elev}$ is shown in Figure 4.6b, together with the point cloud C_s, presented in Figure 4.6a.

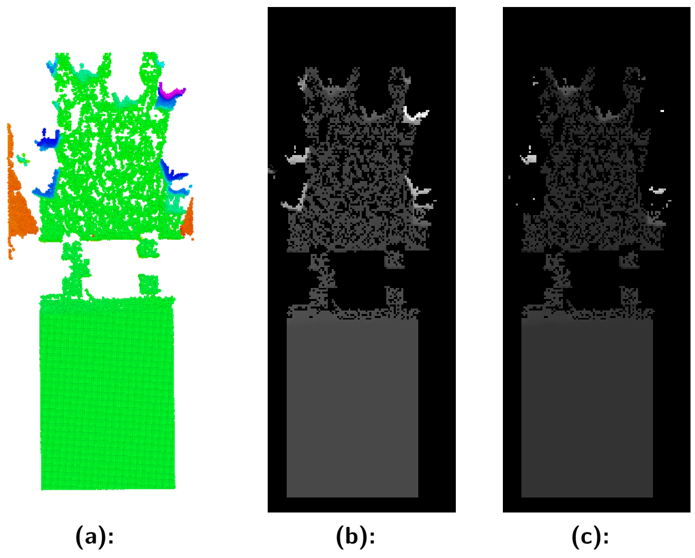


Figure 4.6: Particular steps of the creation of the elevation map $\mathcal{M}_{elev}$ from the point cloud, where (a) is the point cloud with the RGB color schema with purple as the highest altitude; (b) is the elevation map $\mathcal{M}_{elev}$ with grey color schema (white is the highest); and finally (c) the elevation map after removing the inaccessible areas.

The origin of the point cloud C_s is located in the center of the sensor device. However, the motion planning is done with respect to the center of the robot, therefore, the origin of the elevation map $\mathcal{M}_{elev}$ is aligned with the center of the robot body. A simple translation of the point cloud C_s is applied, due to the considered way the sensor is mounted on the robot.

4.5 Foothold map

Not all the locations represented in the elevation map $\mathcal{M}_{elev}$ are accessible, due to the dimensions of the real robot body. Therefore, a foothold map B is constructed to define suitable foothold locations of the elevation map $\mathcal{M}_{elev}$. The foothold map is in a form of binary image that divides the locations to accessible and non-accessible.

The robot is unable to place the endpoint of its leg right next to an obstacle, because of physical limitations considering the size of the endpoint of the robot leg. It can place the endpoint on the obstacle if there is enough free space, but never right next to it, as it would result in a collision of the foothold with the obstacle. In a case of such a situation, the points lying on an edge between two areas of different height, have to be removed. The robot cannot access the space near the edge both from the lower area, resulting in a collision with the object, nor from the higher area, resulting in a limited support. Result of such an approach is presented in Figure 4.6c.

Having removed areas lying near the edge of an object, it is necessary to filter areas the robot cannot reach, e.g., a bottom of a deep gap. These areas are defined by the dimensions of the robot body, together with the forward robot motion in which the robot keeps the body horizontally to the supporting ground. The robot is able to enter an area with the maximum depth of 0.25 metres from the robot body; thus, a deeper area is considered as inaccessible, and therefore, it is removed from the possibly feasible locations.

A similar approach is performed when filtering out points with undesirable normals, as we are looking for locations with the maximal support. When considering a planar motion, the maximal support to the robot stable position is provided by the points with normals perpendicular to the plane and thus, points with normals with the deviation greater than $\pi/4$ from the normal of the horizontal plane are considered as unstable and marked as non-accessible in the foothold map. The deviation ψ is calculated using the formula

$$\psi = \tan^{-1}\left(\frac{\|\vec{v} \times \vec{n}\|}{\vec{v} \cdot \vec{n}}\right), \tag{4.4}$$

where $\vec{v}$ is a normal of the particular point and $\vec{n}(0,0,1)$ is a vector corresponding to the normal of the horizontal plane.

The unconstrained areas of the terrain provide enough locations for the foothold placement. However, the limited passage in front of the robot, provide only a limited number of locations. Therefore, a search for a suitable location is speeded up by an evaluation function that enables the robot to select the most suitable location for the foothold placement based on the foothold map. The proposed evaluation function is described in the next section.

4.6 Evaluation function

The proposed evaluation function F is considered to represent a robustness of each bin in the foothold map B. The point cloud provided by the camera is not dense enough to fill every bin of the elevation map. Therefore, a number of vacant bins are in the elevation map and consequently in the foothold map, which decreases the number of places available for a foothold selection. The main idea of the proposed evaluation function F on a single pillar is shown in Figure 4.7.

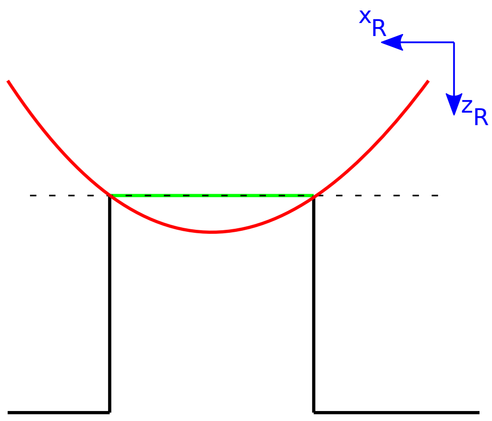


Figure 4.7: Evaluation function of the accessible locations in xy plane where the terrain is in black, evaluation function is in red, accessible position is in green and dashed line represents locations where evaluation function crosses zero.

The real terrain in front of the robot is considered to be a continuous space; therefore, vacant bins in a close vicinity of an occupied bin are assumed to be accessible in the real terrain. As the foothold map is a binary image, the gaps can be corrected using methods of the mathematical morphology [32]. Using structure element smaller than the real robot endpoint, smaller holes in the real terrain can be omitted, while the larger ones remain. Therefore, a square with dimension 5×5 cm (5×5 pixels) is considered as the structure element. The first applied operation of mathematical morphology is dilation, which is a union of translated point sets

$$X \oplus S = \bigcup_{s \in S} X_s, \tag{4.5}$$

where X is a binary image and S is the structure element [32].

Whenever a particular binary pixel in the image X is marked as occupied (accessible), its neighbourhood equal to the structure element with the particular pixel in its center is also marked as occupied in the resulting image; so, areas of the occupied pixels are expanding. The result of such an operation applied on the elevation map is shown in Figure 4.8b.

The second applied operator is erosion, which is an intersection of all image X translations by the vector $-s \in S$.

$$X \ominus S = \bigcap_{-s \in S} X_{-s}, \tag{4.6}$$

where X is a binary image and S is a structure element [32]. If the neighbourhood of a particular pixel marked by the structure element S consists only of occupied pixels then, the particular binary pixel is marked as occupied;

otherwise it is marked as vacant. Dilation followed by erosion is called binary closing and can be formally defined as

$$X \bullet S = (X \oplus S) \ominus S. \tag{4.7}$$

Binary closing removes smaller gaps possibly created by faulty measurements and preserves greater ones placed in front of the robot. An example of the binary image B_c, created by the binary closing operation is shown in Figure 4.8c.

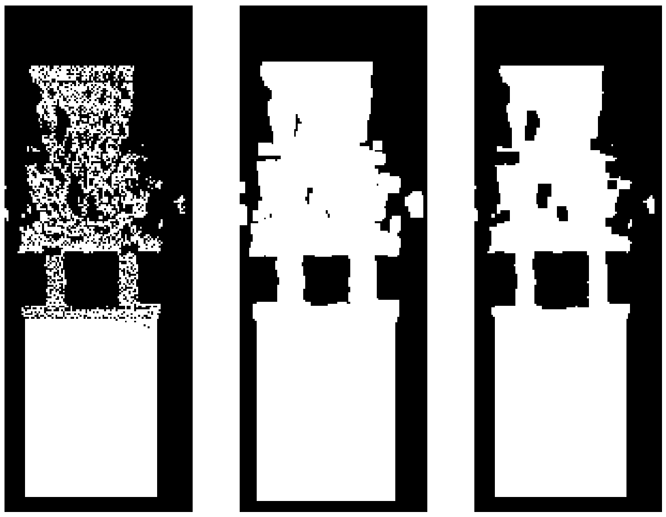

(a) : Foothold map B **(b)** : Dilation **(c)** : Binary image B_c

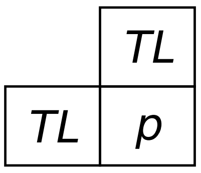
Figure 4.8: Operations over binary image where accessible areas are in white and non-accessible areas are in black.

A value that represents the distance from the nearest non-accessible pixel is calculated for every pixel of the binary image B_c; therefore, a distance transform algorithm is applied. Input of the distance transform algorithm is a binary image, where pixels are either occupied (accessible) or empty (non-accessible). Using a local mask of the neighbouring pixels [33], a minimal distance between each occupied pixel and the nearest empty pixel is calculated. The Manhattan distance is applied, which calculates the distance between the pixels p and q, using formula

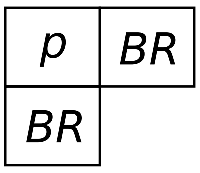
$$D(p, q) = |p_x - q_x| + |p_y - q_y|, \tag{4.8}$$

where (x, y) denotes the position of the particular pixel within the binary image B_c.

A two-pass algorithm [34] is applied to determine evaluation function that assess suitability of particular foothold location. In the first pass, the top-left (TL) mask, presented in Figure 4.9a, is iteratively applied to all pixels, going



(a) : Top-left mask



(b) : Bottom-right mask

Figure 4.9: Local masks for distance transform

from the top-left corner of the binary image down in a row-wise manner from left to right. An updated value is calculated for every pixel p using formula

$$I(p) = \min_{q \in AL} (I(p), D(p,q) + I(q)), \tag{4.9}$$

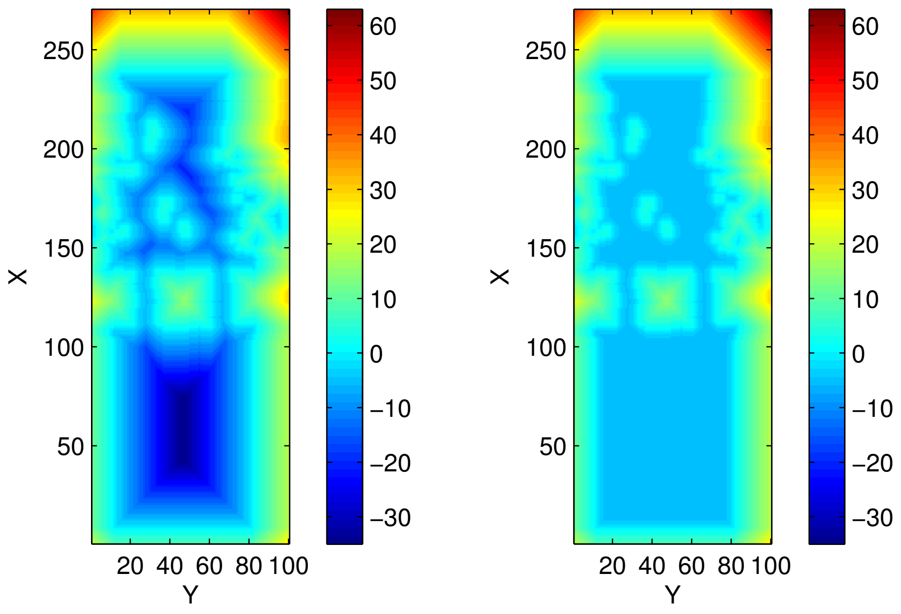
where I is the resulting image describing the evaluation function being determined.

For the second pass, the bottom-right (BR) mask, see Figure 4.9b, is iteratively applied on all pixels of the binary image I, starting from the bottom-right corner, going up through the binary image in a row-wise manner from right to left. An update value is again calculated for every pixel p using the formula

$$I(p) = \min_{q \in BR} (I(p), D(p,q) + I(q)). \tag{4.10}$$

This approach is analogically applied for the inversion of the binary image B_c, calculating the distance between particular pixel and the nearest accessible pixel, which results in the distance transform of the inversion of the binary image B_c stored in the image I_I. Therefore it is possible to track the nearest accessible pixel for a particular non-accessible pixel.

Concatenating the results of both approaches (I and I_I), an evaluation function F is created, depicting the pixels with the maximal estimated support as the local minimum of the function. Therefore, the distance values in the image I are treated as negative. It is then possible to search for a local minimum, every time a non-accessible pixel is selected, which speeds up the planner. An example of the evaluation function F is shown in Figure 4.10a. However, locations deeper than 5 cm inside an accessible area are considered equally suitable for foothold placement, therefore, the function is thresholded by a bound equal to -5, see the idea in in Figure 4.11. An example of the thresholded evaluation function F_t is in Figure 4.10b. Areas with the zero value in the evaluation function represent the edges between accessible and non-accessible areas.



(a) : Evaluation function F **(b) :** Thresholded evaluation function F_t

Figure 4.10: Evaluation function of a foothold accessibility

4.7 Troubleshooting

Based on our early experiments, the measured point cloud is not sufficient to provide a consistent terrain reconstruction, as it is shown in Figure 4.12a. There is a physical limitation, considering the functionality of the ASUS Xtion PRO, such as the minimal required sensing distance to obtain reliable distance measurements. The device is unable to perceive terrain closer than 0.5 metres, therefore, it does not only influence the perception of physical obstacles located in front of the robot, but mainly causing the robot unable to sense the terrain that it should traverse.

Fortunately, one of the advantages of the six-legged robots is its modality considering its body configuration. While the robot is unable to perceive the terrain when it is in the walking state close to the surface (e.g., see Figure 4.13a), the situation changes when the robot moves the body into an upper configuration (see Figure 4.13b) where the depth camera is placed higher above the terrain (approximately about 7 cm higher). The perception of the terrain is then significantly better as it is shown in Figure 4.12b.

Therefore, the measurements taken from the upper configurations are used for the terrain modelling. Even in the walking state, the robot is able to perceive parts of the terrain, but mostly objects rather than terrain; so, these measurements are more suitable for the motion estimate than for the terrain

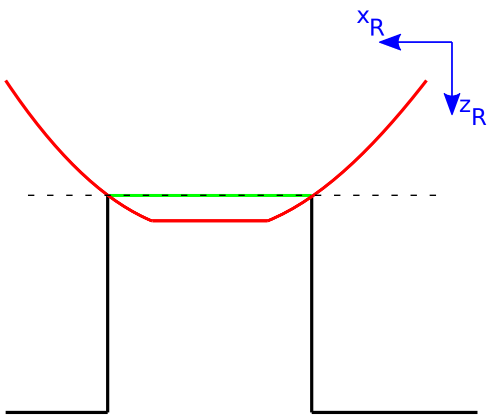
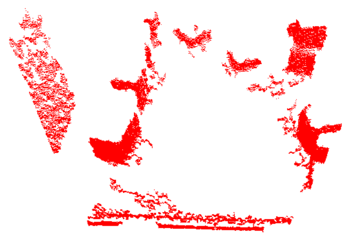


Figure 4.11: Thresholded evaluation function of the accessible locations in xy plane where the terrain is in black, the evaluation function is in red, accessible position is in green and dashed line represent locations where evaluation function crosses zero.

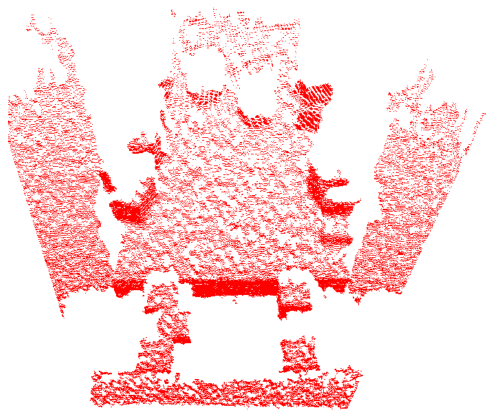
reconstruction.

The extra vertical motion for taking measurements of the terrain has to be explicitly considered in the planning, because the body position with the maximum height cannot be achieved from all regular walking configurations due to the kinematic margin.

The second problem concerning the limitation of the camera is the initialization of the world model. The closest area the robot is able to perceive is located at least 0.5 metres in front of the robot; therefore, it is assumed the robot initial location is on a flat surface and the world model is initialized with the filled area between the robot initial position and the closest perceived obstacles. The initial area beneath the robot is modelled as a plane as shown in Figure 4.14. Once the robot moves forward, the initial area is amended by real measurements, gained from the camera.

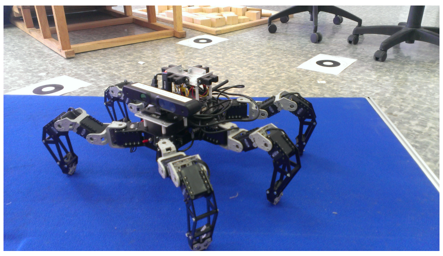


(a) : Perception from lower walking position

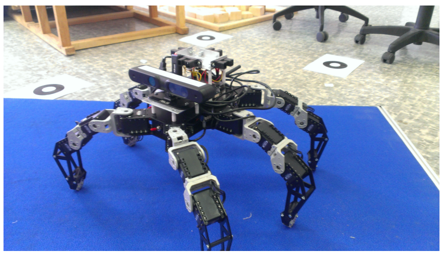


(b) : Perception from upper position

Figure 4.12: Perception of the surface regarding the body height



(a) : Robot in lower walking position



(b) : Robot in upper position

Figure 4.13: Body height adaptation

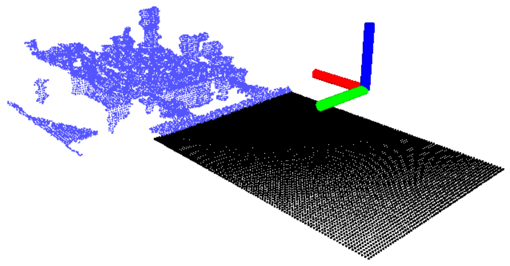


Figure 4.14: Initialization of the world model where measurements provided from the camera are in blue, the initial terrain model is in black and the center of the camera is represented by the 3-axes origin of the coordinate system.

Chapter 5
Motion Planning

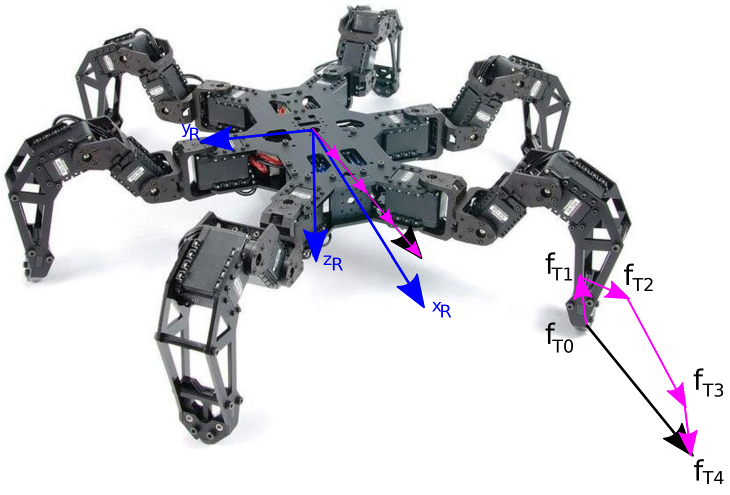


Figure 5.1: The proposed idea of the decomposed motion planning is based on split motion planning of the robot body followed by sampling possible foothold location and local planning of the leg motion. The expected motion steps are in magenta and the robot coordinate frame is in blue.

The proposed motion planning approach is based on incremental construction of a motion planning roadmap that can be represented as graph $G(V, E)$, where each node $q_a \in V$ represents a configuration and an edge $e \in E$ connecting two nodes $e = (q_a, q_b), q_a, q_b \in V$ represents a feasible motion. Since the motion planning is started from the given robot configuration, the roadmap graph G is a tree and the initial robot configuration is the root of the tree q_0. Each node contains information about the robot configuration (absolute position of the robot body and angles of all joint actuators), previous node and a score represented as the weight w_a, which denotes suitability of the locations as a supporting foothold. Since a general motion planning for six-legged walking robot is computationally challenging and we aim to provide real-time solution, the proposed roadmap expansion is based on randomized sampling of the possible body movement followed by sampling of new foothold location for moving leg in the used pentapod gait. Then, feasibility of such a new configuration is evaluated and leg motion is refined according to the swing motion of the leg motion step. Besides, local search techniques are employed to find the most suitable body motion and foothold placement considering the elevation map $\mathcal{M}_{elev}$, evaluation function F and the weight w to provide robust and safe path to traverse the passage with only few foothold locations. An example of the proposed motion is in Figure 5.1.

The motion planning follows the idea of the incremental roadmap construction as in the randomized sampling-based motion planners [35] that is combined with a deterministic expansion utilized in the lattice based approach [36]. The main idea of the proposed planning approach is based on expanding the initial robot configuration considering an independent sampling of the possible body movement with consecutive sampling of the next foothold locations. Thus, for a single sampling of up to m body movements, up to n foothold locations defined by neighbourhood of a regular motion gait for the particular leg are sampled and feasibility of such motion is evaluated using IKT and collision check. Then, the most suitable locations according to the weight is selected and the process is repeated for the next leg motion in the gait. Finally, the whole planning is repeated up to the given l number of gait cycles, where the used gait prescribes which leg is considered for the new foothold placement.

The search is terminated after l gait cycles the robot travels a distance longer than Δx while still keeping the straight direction and its change in the y-axis is below Δy, because we assume the passage to be traversed is straight ahead of the initial robot location and it is shorter than Δx. The expected goal position of the robot is defined in the distance Δx ahead of the current robot position. Particular sampling of the next robot location and new foothold together with the motion planning for each individual leg and body are described in the following sections.

Notice, once a motion plan is determined and executed, the robot takes a new sensor measurements to update the terrain model and the planning is repeated for the current robot position. Besides, sensor measurements are collected during the robot motion to update the point cloud map and support the global localization of the robot.

5.1 Robot body motion

The robot is standing on five legs and the remaining one performs the forward step; the five standing legs are deterministically driven by the motion of the robot body. Therefore, the control of footholds of the five standing legs, i.e, 15 actuators, can be performed by two coordinates representing the robot body motion (x, y), which reduces the 15 DOF down to 2 DOF.

The proposed motion planning of the robot body considers up to m random movements, regarding the distance and angle, to generate a variety of robot body positions from which the foothold locations for the moving leg are searched. The expected robot body motion forms a cone with the origin at the center of the robot coordinate system and pointed towards the passage to be traverse. The expected area from which the body locations are sampled is visualized in Figure 5.2.

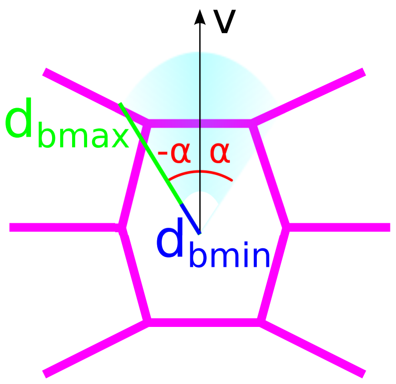


Figure 5.2: Expected motion of the robot body is in cyan where the probability of selecting a particular motion is represented by its transparency.

The next robot body location is sampled within the cone defined in the the angle α. Besides, it is expected the next body center should move by at least d_{bmin} distance and not far than d_{bmax}. Therefore, the intended traversed distance of the body is defined using bounds d_{bmin} and d_{bmax} as

$$d_{rand} = d_{bmin} + (d_{bmax} - d_{bmin})rand(), \tag{5.1}$$

where $rand()$ is a function that provides a random uniform distribution in the range $\langle 0, 1\rangle$. The angle of the robot movement is defined as

$$\phi_{rand} = sample(\alpha), \tag{5.2}$$

where $sample(\alpha)$ is a function that provides a random value in the range $\langle -\alpha, \alpha\rangle$ with the Gaussian distribution. Thus, the expected new body position is defined by d_{rand} and ϕ_{rand} .

Notice, only two coordinates are sampled as the considered forward motion is performed at the constant height and the standing legs are supposed to remain on the ground.

5.1.1 Feasibility tests

Once the body motion is defined, configurations of the five standing legs are subject to IKT described in Chapter 3. The endpoints are forbidden to be set in the area under the robot body, due to the lack of support; furthermore, the endpoints of the front legs are forbidden to be set behind the front level of the robot body due to the consequent change of the robot center of mass towards the frond part of the robot body. Based on experimental evaluation, such a change of the center of mass results in an imbalance of the robot body. If a new configuration does not pass the considered tests, the proposed robot body motion is rejected and a new sample of the robot body location is selected.

A collision between the legs is not considered in this section as all five legs move towards same direction defined by the new body location and a collision of the leg with the body is eliminated by the actuator limits defined in Chapter 3.

The robot body motion influences the leg that performs the forward step; therefore, the body motion is calculated first and consequently, the foothold of the leg that performs a forward step is selected.

5.2 Foothold selection

The foothold position selection for the stepping leg, is defined analogically to the robot body motion by assuming up to n random samples of the new foothold location, see Figure 5.3. However, different parameters defining the bounds of the traversed distance, d_{lmin} and d_{lmax}, of the stepping leg are considered; and thus, a random sample is drawn from a cone defined by the distance d_{rand} and angle ϕ_{rand}

$$d_{rand} = d_{lmin} + (d_{lmax} - d_{lmin})rand() \tag{5.3}$$
$$\phi_{rand} = sample(\beta), \tag{5.4}$$

where $rand()$ and $sample(\beta)$ are analogical to the functions used for the body motion.

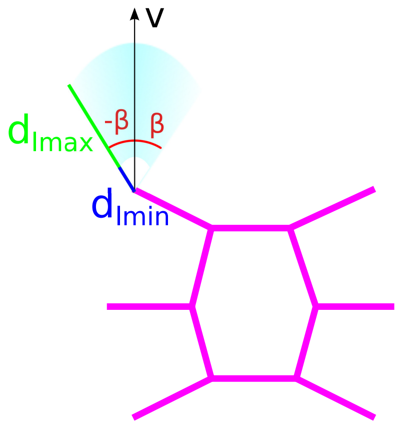


Figure 5.3: Motion of the leg that performs a forward step is in cyan where the probability of selecting a particular motion is represented by its transparency.

Having a new location of the leg endpoint in the global coordinate frame, the expected height of the foothold is determined from the elevation map $\mathcal{M}_{elev}$.

5.2.1 Feasibility tests

Once the foothold position is defined, it is checked whether the foothold is accessible regarding its location in the evaluation function F. If the value of the location in the evaluation function is strongly negative, the location corresponds to the accessible area and the foothold is accepted. In a case, the value in the evaluation function is negative, but close to zero, or positive, the foothold is not immediately rejected. A local greedy search is performed to find the closest local minimum of the evaluation function F.

The intended foothold is analysed by the IKT to test if the corresponding configuration of the leg is in a reachable position. Analogically to the body motion, it is forbidden to set the endpoints in the area under the robot body due to the lack of support. Besides, it is forbidden to set the endpoints of the rear legs in front of the rear level of the robot body due to the consequent change of the robot center of mass towards the rear part of the robot body.

Moreover, the intended leg endpoint cannot be located in a close vicinity of the other current endpoints due to a possible collision. The construction above the rubber tip of the leg is the cause of possible collisions and has to be considered to eliminate the colliding configurations. Therefore, configurations in which the endpoints of two legs are closer than 6 cm are rejected.

Once the foothold is indicated as feasible, a candidate configuration q_a for the next robot step is created and considered for further evaluation according to its weight and the proposed roadmap expansion criterion described in the next section.

5.3 Expansion criterion

An expected stability of the configuration is the main factor considered in this thesis; therefore, it is the strongest element of the proposed criterion to select a configuration for the roadmap expansion. However, the forward body motion should also be considered regarding the selection of the next node for the expansion. Therefore, the weight of a particular node is defined as

$$w = F(x_{l_i}, y_{l_i}) - \left(\frac{d_g}{d}\right), \tag{5.5}$$

where (x_{l_i}, y_{l_i}) denotes the location of the currently moving leg l_i in the global coordinate system, F is the evaluation function (see Section 4.6), d_g is the current distance of the robot to the expected goal location, and d is the expected total travelled distance from the robot initial location to the goal location.

Having a set of candidate foothold locations S for the current leg l_i, the further expansion is performed from the most safe location q_a that is found as

$$q_a = \arg\min_{q_i \in S} w(q_i), \tag{5.6}$$

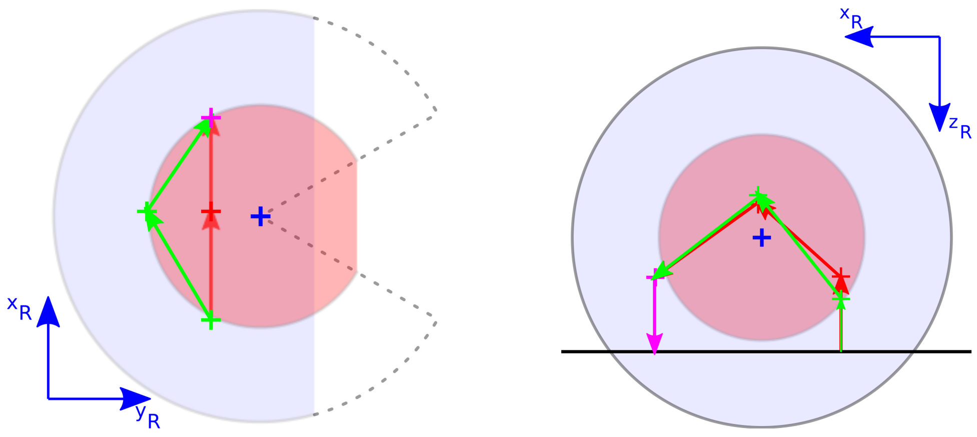
where $w(q_i)$ is the weight assigned to node q_i.

Once the current tree is expanded in such a way that the robot can be moved about distance Δx towards traversing the passage, a sequence of the motion is constructed from the roadmap using backtracking. However, the sequence cannot be directly applied due to the simplification utilized in the planning of the foothold selection for the forward motion of the moving leg. Therefore, the sequence is further processed to create a detailed trajectory with intermediate configurations for lifting the leg up and a precise determination of the joint angles.

5.4 Trajectory generation for the leg forward motion

The main idea of the trajectory generation is to provide a detail sequence of the intermediate steps T of the moving leg to avoid possible collision with obstacles during the swing phase. Therefore, the motion of the leg from one foothold location to a new one is split into 3 intermediate configurations defined by f_{T1}, f_{T2}, f_{T3} that are shown in Figure 5.1.

5.4.1 Feasibility tests



(a) : Top view of the operational space

(b) : Side view of the operational space

Figure 5.4: Adjusting the trajectory of the stepping leg regarding its operational space where non-reachable space is in semi-transparent red, the operational space of the leg is in semi-transparent blue, the origin of the leg is represented by blue cross and the terrain is represented by black line. The approved motion is in magenta, the rejected default trajectory is in red and the approved trajectory is in green.

Every configuration represented in the trajectory is tested by the IKT to define its reachability. In a case the endpoint is not indicated as reachable, the nearest reachable position in the leg operational space is provided with respect to the Euclidean distance. An example of the refined trajectory is shown in Figure 5.4.

In addition to the test performed by the IKT, the second test is performed for the point f_{T2} considering the surface beneath the stepping leg. The leg endpoint should not be drawn near the terrain except the positions f_{T0} and f_{T4} that are supposed to provide the only contact with the terrain. Therefore, the second and third motion of the trajectory T are projected to the elevation map and the minimal distance between the particular trajectory section and its projection is estimated. If the distance is beneath the allowed level of 2 cm, the nearest higher reachable endpoint position in the leg operational space is selected. The selection process is repeated until a position satisfying all conditions is found. An example of the collision evasion is presented in Figure 5.5.

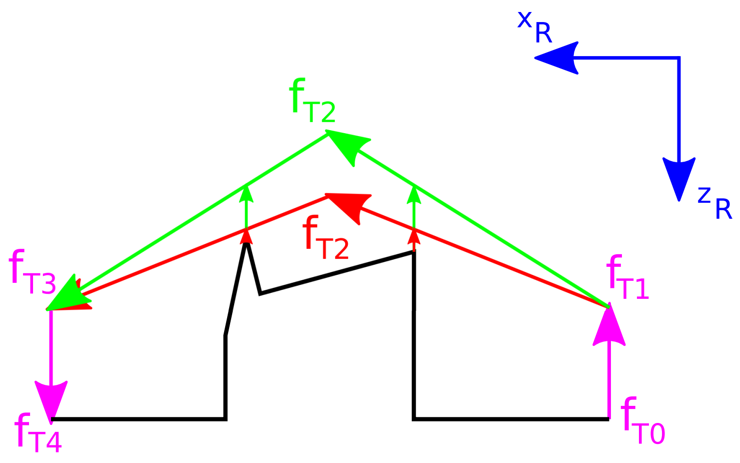
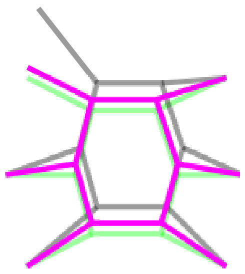
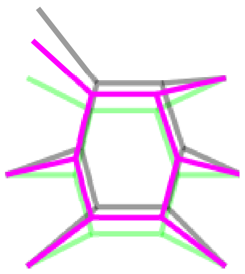


Figure 5.5: Proximity hold in xz plane where the terrain is represented by black, the accepted motion is in magenta, the colliding default trajectory is in red and the refined trajectory is in green.

Three robot configurations are created based on the refined trajectory T, which defines a motion primitive that is intended to improve robustness by avoiding slippage during the motion of body and legs. The resulting sequence of smooth motion with left front leg performing a forward step is shown in Figure 5.6.

Once the final trajectory as a sequence of feasible configurations is determined, it is directly applied to the robot control and performed. Then, the robot collects new sensor measurements by moving the body into the upper configurations, data are merged into the world model and the whole process is repeated until the robot passes the passage.


(a) : Configuration with f_{T1}


(b) : Configuration with f_{T2}

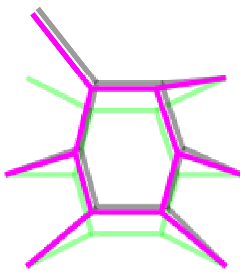

(c) : Configuration with f_{T3}

Figure 5.6: Five consecutive configurations of the robot motion in xy plane, where the configuration q_b is in green, the configuration q_a is in black, and configurations with with the front left leg performing a step are in magenta.

Chapter 6

Experimental Results

The proposed approach has been verified in a laboratory testbed that provides realistic scenario for experimental evaluation. The testbed consists of passage between two flat surfaces. The gap in the passage is bridged by pillars that support with relatively narrow area for foothold locations. We consider various scenarios regarding the number and position of the pillars to evaluate performance of the proposed approach in different scenarios, see Figure 6.1.

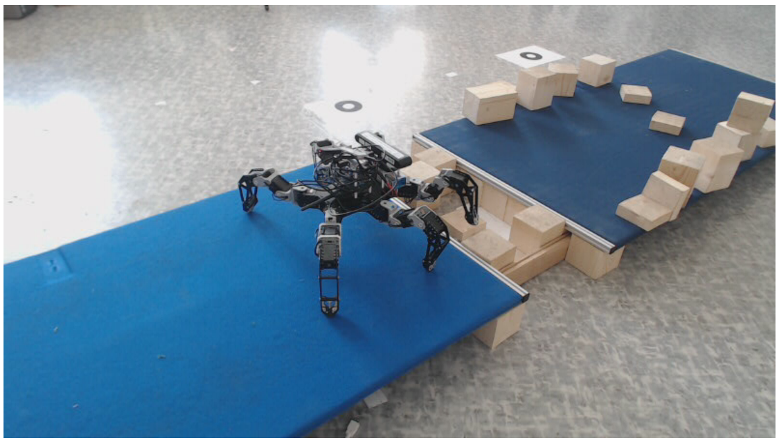


Figure 6.1: Proposed scenarios

In each scenario, the robot is placed in front of the passage in a distance that allows the perception of the passage, regarding the limitation of the applied sensor. Three trials are assumed for each scenario and a trial is considered as successful, if the robot passes the passage without falling down and the whole body of the robot is located on the flat surface behind the gap, i.e., the robot passes the passage.

The robot needs to plan the motion three times to successfully traverse the passage regarding the considered distances of the testbed. In this section, we report on experimental evaluation of the method and for each scenario the reconstructed terrain representation with foothold map is presented. Performance of the proposed solution is reported for each scenario and indicators of the motion planning are presented for all locations, trials, and scenarios.

6.1 First scenario

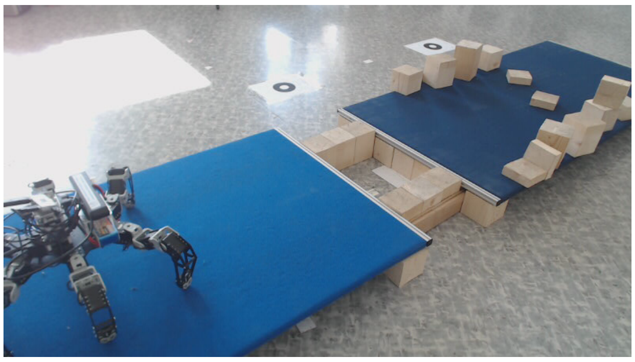
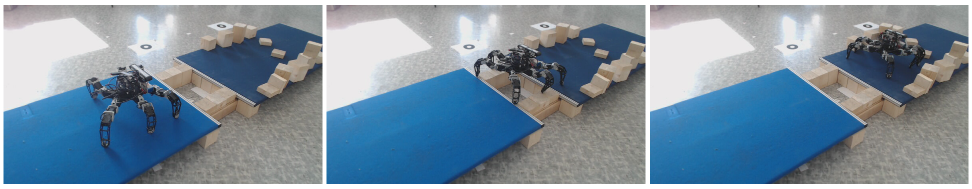


Figure 6.2: Initial robot configuration in the first scenario

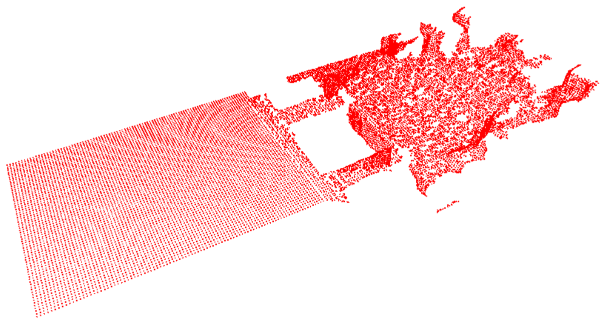
The first scenario represents the easiest form of the passage; it consists of 6 pillars divided into 2 groups of 3 pillars. The groups are placed in a straight line with a gap between them to allow the robot motion; thus, forming two bridges between the unconstrained areas. The first scenario is presented in Figure 6.2.



(a) : Second location **(b) :** Third location **(c) :** Terminal location

Figure 6.3: Traverse of the first scenario

The process of traversing is divided into three sections in which the robot consecutively plans its motion. The robot at the particular planning locations is shown in Figure 6.3. The terrain reconstruction in the first scenario is presented in Figure 6.4a and the foothold map is shown in Figure 6.4b. It was visible that the robot preferred footholds in the centre of the particular bridge, due to the applied evaluation function, which indicates the centres as the locations with maximal robustness.


(a) : Terrain reconstruction of the first scenario

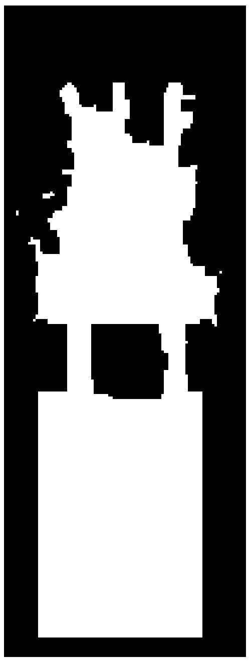

(b) : Foothold map of the first scenario

Figure 6.4: Terrain modelling of the first scenario

6.2 Second scenario

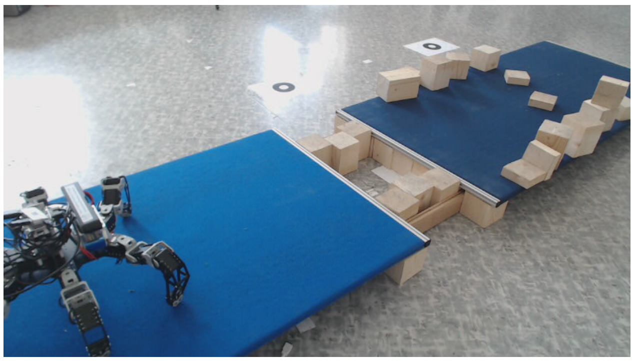
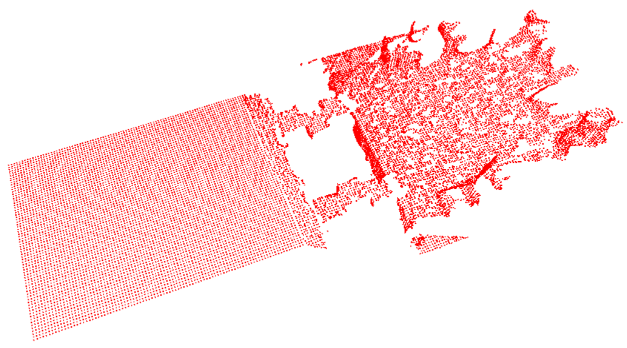


Figure 6.5: Initial robot configuration of the second scenario

The second scenario is similar to the first scenario; again 6 pillars in 2 groups are used, but the middle blocks are shifted more to the center of the passage to test the adaptation of the approach. The second scenario is presented in Figure 6.5 with the robot placed in the initial position.

The terrain reconstruction in the second scenario is presented in Figure 6.6a and the foothold map is presented in Figure 6.6b.


(a) : Terrain reconstruction of the second scenario

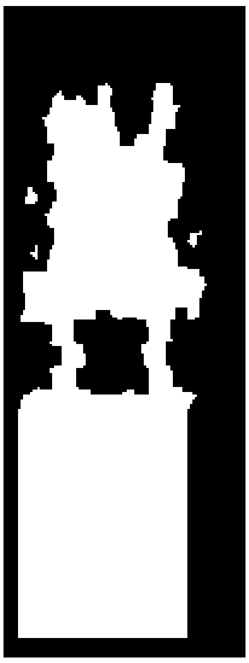
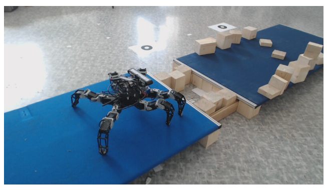
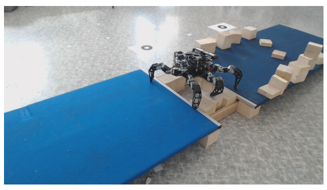

(b) : Foothold map of the second scenario

Figure 6.6: Terrain modelling of the second scenario

The three locations in which the robot plans its consequent movement are presented in Figure 6.7 and the terminal location of the traverse is presented in Figure 6.7c.


(a) : Second location


(b) : Third location

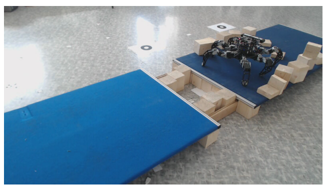

(c) : Terminal location

Figure 6.7: Traverse of the second scenario

6.3 Third scenario

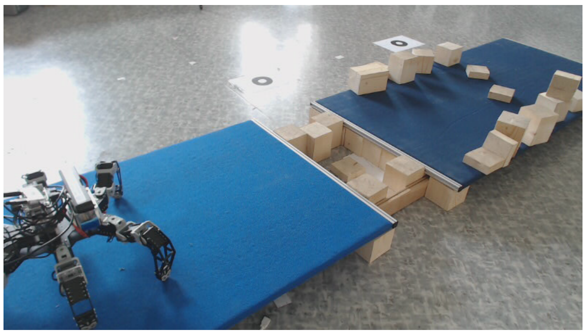
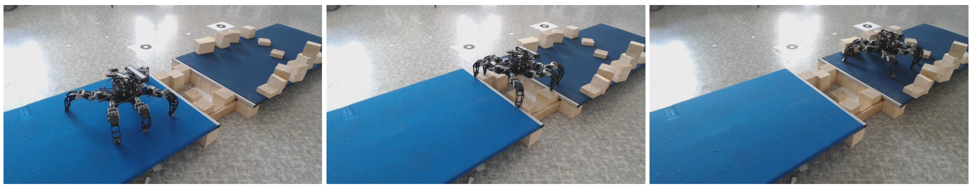


Figure 6.8: Initial robot location in the third scenario

The third scenario consists of 5 pillars in 2 groups, where one group forms a sequence of 3 pillars with the middle pillar shifted as in the second scenario. The second group is formed by 2 pillars with a gap in between them, forming a more difficult passage. The third scenario is shown in Figure 6.8 with the robot placed at the initial position.

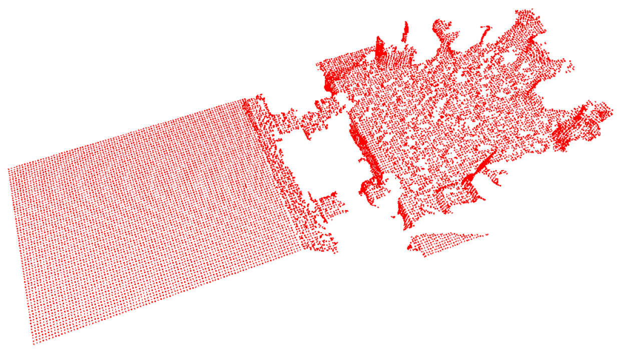


(a) : Second location **(b) :** Third location **(c) :** Terminal location

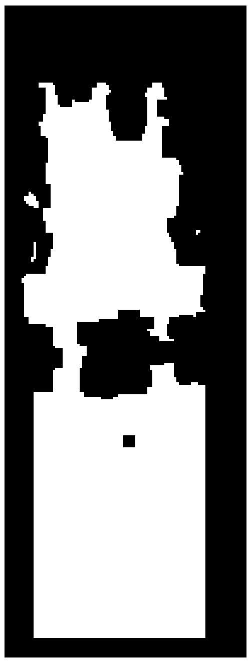
Figure 6.9: Traverse of the third scenario

The planning locations for the consequent movement are presented in Figure 6.9 and the terminal position is presented in Figure 6.9c. The terrain reconstruction in the third scenario is presented in Figure 6.10a and the foothold map is presented in Figure 6.10b.

The gap in the third scenario is difficult for the motion of the robot; therefore, a specific sequence is applied. The leg that is about to traverse the gap has to maintain on the pillar in front of the gap as long as possible while the robot body moves forward. As the origin of the leg moves above the gap, the foothold located behind the gap is selected.



(a) : Terrain reconstruction of the third scenario



(b) : Foothold map of the third scenario

Figure 6.10: Terrain modelling of the third scenario

6.4 Performance of the Proposed Motion Planner and Summary of the Experimental Evalution

The robot is able to traverse the considered scenarios in each of the particular trial, according to the plan provided by the motion planner. The performance results are presented in Table 6.1. The planner is always able to generate a feasible plan in less then one second except two sections in the third scenario in which the search takes longer due to the difficult terrain in front of the robot. Nevertheless, the planner is still able to generate a feasible plan. The difficulty of the third scenario is in the 10 cm gap located in the right part of the passage that requires a very specific motion sequence to be provided by the planner.

The foothold locations provided in the plan for the robot motion are corresponding to the locations accessed by the real robot, based on the gathered data about the plan and performed motion. The performed robot motion is statically stable, as expected from the provided motion plan, furthermore, foothold locations in the centres of the pillars were preferred in the robot motion over the passage.

Table 6.1: Performance of the motion planning in three scenarios where n_g represents the total number of nodes in graph, n_e represents the number of expanded nodes and t represents the total computational time to find a feasible path in a particular section of a trial in each scenario

		First scenario			Second scenario			Third scenario		
Trial	Section	n_g	n_e	t $[s]$	n_g	n_e	t $[s]$	n_g	n_e	t $[s]$
1	1	1111	71	0.117	666	71	0.089	933	77	0.104
	2	776	92	0.082	790	122	0.112	1267	1069	0.270
	3	644	222	0.099	936	294	0.147	3715	2380	1.416
2	1	1184	81	0.113	611	74	0.088	1092	83	0.117
	2	1260	224	0.171	703	73	0.076	2943	2407	0.743
	3	781	56	0.082	503	137	0.068	3300	2146	1.151
3	1	533	68	0.076	841	90	0.097	1133	118	0.140
	2	445	102	0.057	429	148	0.070	3301	2669	0.964
	3	464	95	0.049	827	140	0.104	1083	457	0.214

6.4.1 Discussion and Possible Future Work

Although the proposed approach provides a feasible solution of the considered problem and the robot successfully passes all the considered scenarios, there are numerous ways how it can be improved. Few of the ideas are described in the section to provide a ground for future work towards autonomous navigation in challenging environments.

Due to the nature of the six-legged walking robots, the statical stability can be considered for the purpose of the robot motion. The robot is able of using three legs for supporting the robot body, while keeping the body in a stable position. Planning the robot motion considering only the statical stability limits the robot from entering configurations that results in an imbalance of its body. This approach allows the robot to stop in every moment of its motion, because the center of gravity of the robot is still located near the center of the robot body. Therefore, the robot can choose any foothold placement as long as the condition for the stability maintenance is satisfied.

The obvious disadvantage of this approach is the limitation considering the number of possible configurations. The considered requirement to keep the robot in a statically stable position, rejects configurations in which the robot is on the border of stability and can be overcome by dynamic motion or a suitable more complex motion trajectory. A dynamic model of the robot is substantial for solving such a task, incorporating forces applied to the actuators, friction of the robot foothold with the ground and gravitational force applied to each part of the robot. More complex motion can be achieved with such a robot model, e.g., climbing over obstacles exceeding the robot dimensions or performing considerable leaps impossible in lower speed; hence, extending the overall robot motion capability.

Another extension of the proposed approach is an integration of proprioceptive sensors. The robot actuators can act as sensors that provide additional information about the terrain. Such a sensor can compensate the disadvantage of the camera regarding the minimal perception distance; thus, any error caused by slippage or wrong foothold placement is immediately sensed and the robot can take action to deal with such a situation. The reactive behaviour of such an extension also enhances the information about the terrain, as the measurement of the surface, taken by the camera, does not provide information about the adhesive property of the surface, e.g., waxed floor, sand, etc.

Similar scenario appears when the robot uses a top surfaces of the obstacles for traversing that can be perceived as stable in measurements, but becomes unstable in reality because of the applied robot action force. The particular object is stable when its center is selected for foothold placement, but becomes unstable when a position near its edge is used for the same purpose, e.g., thin high block, pillar,etc. In such a case, the actuator senses the lack of support from the terrain and an alternative plan for the robot is generated.

Chapter 7

Conclusion

The motion capabilities of the six-legged walking robots are considered in this thesis. The flexibility regarding its body configuration allows the robot to traverse a kind of surface that no other ground robot is capable of. Such a forward motion requires precise information about the immediate surrounding and the robot kinematic model. Based on these two factors, a sequence of motion can be established, resulting in a forward motion of the robot.

The ASUS Xtion PRO camera is considered as a suitable off-the-shelf low-cost sensor that provides a point cloud representing the real terrain in front of the robot. As the robot moves forward, new measurements are integrated into the world model of the robot surroundings and a consistent precise world model of the robot surroundings is constructed. Furthermore, the robot localizes itself with respect to the terrain in front of it simultaneously with the model construction.

A grid based elevation map is selected as a suitable structure for motion planning and the world model is converted into the elevation map. Based on the elevation map, a foothold map is created, in which accessibility of the particular bins of the elevation map is indicated, and finally an evaluation function is defined to assess suitability of the locations for the foothold placement.

The motion planner accesses the evaluation function and elevation map, to find a feasible sequence of robot configurations that allow the robot to traverse the passage with limited number of footholds. The sequence of configurations is subject to the defined kinematic model that allows only stable, reachable and non-colliding configurations of the robot body.

The performed experimental verification of the proposed approach confirmed its usability in various scenarios. Furthermore, it verified its ability to adapt to various situations in which the proposed motion planner provides a sequence of configurations for a stable forward motion. The performance results of the motion planner correspond to the difficulty of the particular terrain, regarding the number of expanded nodes to find a feasible path. The first section of each scenario is very similar, and therefore, similar results are provided. Moreover, the first section of each scenario is supposed to be the easiest to traverse, which is reflected in the measured performance indicators of the proposed motion planner.

Regarding a possible extension of the proposed algorithm, the robot actuators can be utilized as the proprioceptive sensors. Their employment enhances the information about the terrain, concerning its friction and stability of objects, i.e., features that cannot be directly perceived by the camera. Furthermore, the dynamical model of the robot would be a great support towards extension of the robot motion possibilities, e.g., obstacles exceeding the robot dimensions can be traversed only using a dynamic motion. Moreover, it would enhance exploitation of the operational space of the robot legs, which can provide a better overall robot motion.

Bibliography

[1] Y. Tanaka, Y. Ji, A. Yamashita, and H. Asama, “Fuzzy based traversability analysis for a mobile robot on rough terrain,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3965–3970.

[2] H. Lu, J. Gao, J. Zhu, Z. Xu, H. Cao, J. Zhao, F. Zhao, X. Li, Y. Liu, and X. Shi, “Unknown environment 2-d map building system for coal mine detect robot,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 450–455.

[3] K. Ohno, S. Kawatsuma, T. Okada, E. Takeuchi, K. Higashi, and S. Tadokoro, “Robotic control vehicle for measuring radiation in fukushima daiichi nuclear power plant,” in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2011, pp. 38–43.

[4] L. Xu, W. Liu, Z. Wang, and W. Xu, “Gait planning method of a hexapod robot based on the central pattern generators: Simulation and experiment,” in *International Conference on Robotics and Biomimetics (ROBIO)*, 2013, pp. 698–703.

[5] J. Mrva and J. Faigl, “Tactile sensing with servo drives feedback only for blind hexapod walking robot,” *RoMoCo*, pp. 240–245, 2015.

[6] D. Belter, P. Labecki, and P. Skrzypczyński, “An exploration-based approach to terrain traversability assessment for a walking robot,” in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2013, pp. 1–6.

[7] D. Belter, P. Łabęcki, and P. Skrzypczyński, “Adaptive motion planning for autonomous rough terrain traversal with a walking robot,” *Journal of Field Robotics*, vol. 33, no. 3, pp. 337–370, 2016. [Online]. Available: http://dx.doi.org/10.1002/rob.21610

[8] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, “A solution to the simultaneous localization and map building (slam) problem,” *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[9] P. S. D Belter, M Nowicki, “On the performance of pose-based rgb-d visual navigation systems,” in *ACCV*, 2014, pp. 407–423.

[10] G. Klein and D. Murray, “Parallel tracking and mapping for small ar workspaces,” in *6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007, pp. 225–234.

[11] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[12] D. Belter, M. Nowicki, P. Skrzypczynski, K. Walas, and J. Wietrzykowski, “Lightweight rgb-d slam system for search and rescue robots,” *Progress in Automation, Robotics and Measuring Techniques*, pp. 11–21, 2015.

[13] D. Belter and P. Skrzypczynski, “Precise self-localization of a walking robot on rough terrain using parallel tracking and mapping,” *Industrial Robot 40*, pp. 229–237, 2013.

[14] D. Belter and P. Skrzypczyński, “The importance of measurement uncertainty modelling in the feature-based rgb-d slam,” in *10th International Workshop on Robot Motion and Control (RoMoCo)*, 2015, pp. 308–313.

[15] L. Mingxiang and J. Yunde, “Stereo vision system on programmable chip (svsoc) for small robot navigation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 1359–1365.

[16] *ASUS Xtion PRO*, ASUS, (Cited on 10.05.2016). [Online]. Available: http://www.asus.com/Multimedia/Xtion_PRO_LIVE/

[17] D. Masri, “Motion model of stair climbing in hexapod walking robot,” Bachelor Thesis, Czech Techical University in Prague, 2014.

[18] *PhantomX Mark II platform*, Trossen Robotics, (Cited on 09.04.2016). [Online]. Available: http://www.trossenrobotics.com/hex-mk2

[19] ROBOTIS, *Dynamixel AX-12A Features*, (Cited on 09.05.2016). [Online]. Available: http://www.robotis.com/xe/dynamixel_en

[20] *Arbotix Commander*, Vanadium Labs, (Cited on 14.03.2016). [Online]. Available: http://www.vanadiumlabs.com/commander.html

[21] *Arbotix Board*, Vanadium Labs, (Cited on 14.3.2016). [Online]. Available: http://www.vanadiumlabs.com/arbotix.html

[22] *OpenNI*, (Cited on 30.4.2016). [Online]. Available: http://www.openni.org/

[23] *Odroid U3*, Hardkernel, (Cited on 10.5.2016). [Online]. Available: http://www.hardkernel.com/main/products/prdt_info.php?g_code=g13874569627

[24] *Robot Operating System*, ROS, (Cited on 21.5.2016). [Online]. Available: www.ros.org

[25] G. C. Haynes and A. A. Rizzi, "Gaits and gait transitions for legged robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 1117–1122.

[26] P. J. Besl and H. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[27] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10514-012-9321-0

[28] *Moving Least Square*, PCL, (Cited on 10.5.2016). [Online]. Available: http://pointclouds.org/documentation/tutorials/resampling.php

[29] *Point Cloud Library (PCL)*, (Cited on 10.5.2016). [Online]. Available: http://pointclouds.org/

[30] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1991, pp. 2724–2729 vol.3.

[31] *Normal Estimation Algorithm*, PCL, (Cited on 10.5.2016). [Online]. Available: http://pointclouds.org/documentation/tutorials/normal_estimation.php

[32] V. Hlavac, "Mathematical morphology," Lecture, Czech Technical University in Prague, (Cited on 9.5.2016). [Online]. Available: http://cmp.felk.cvut.cz/~hlavac/TeachPresEn/11ImageProc/71-3MatMorpholBinEn.pdf

[33] ——, "Distance transform," Lecture, Czech Technical University in Prague, (Cited on 9.5.2016). [Online]. Available: http://cmp.felk.cvut.cz/~hlavac/TeachPresEn/11ImageProc/014DigitalImageEn.pdf

[34] A. Rosenfeld and J. Pfaltz, "Sequential operations in digital picture processing," *Journal of the Association for Computing Machinery*, October 1966, 13(4):471–494.

[35] S. M. LaValle, *Planning Algorithms.* Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[36] J. Butzke, K. Sapkota, K. Prasad, B. MacAllister, and M. Likhachev, "State lattice with controllers: Augmenting lattice-based path planning with controller-based motion primitives," pp. 258–265.

Appendix A

CD content

The enclosed CD contains ROS nodes source code of terrain reconstruction, motion planning and motion control 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
/map_build	The ROS node for the terrain reconstruction
/cpg	The ROS node for the robot motion planning
/wrlib	The ROS node for the robot motion control
/thesis	The thesis in PDF format
/source	The source files of the thesis