Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

Motion Control of a Hexapod Robot Over Uneven Terrain Using Heightmaps

by

Andries Phillipus Lotriet

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electronic) in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof. J.A.A. Engelbrecht

March 2023

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 2023/02/10

Abstract

Motion Control of a Hexapod Robot Over Uneven Terrain Using Heightmaps

A.P. Lotriet

Department of Electrical and Electronic Engineering,
Stellenbosch University,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MEng (EE)

March 2023

In recent times great strides have been made in the field of autonomous robotics, especially with regards to autonomous navigation of wheeled and arial drones. Legged robotics however still face numerous problems before they can become practical to use, the most egregious of these problems being balancing of the robot and optimal foot placement.

This thesis focuses on providing a solution to the latter problem of foot placement. This is achieved by using an depth camera to, in real time, construct a localised map of the environment and subsequently analysing said map for optimal foot placement locations. The system is then tested using a hexapod robot both in simulation and on a physical robot.

Acknowledgments

Dedication

Table of contents

List of figures

List of tables

List of symbols

Constants

$L_0 =$ 300 mm

Variables

Re_{D}	Reynolds number (diameter)	[ ]
x	Coordinate	[m]
$\ddot{x}$	Acceleration	[m/s^2]
θ	Rotation angle	[rad]
τ	Moment	[N·m]

Vectors and Tensors

$\overrightarrow{\boldsymbol{v}}$	Physical vector, see equation ...

Subscripts

a	Adiabatic
a	Coordinate

Abreviations

IK	Inverse Kinematics	vi
FK	Forward Kinematics	vi
MuJoCo	Multi-Joint dynamics with Contact	2
GUI	Graphical User Interface	7
ROS	Robot Operating System	7
LiDAR	Light Detection and Ranging	6
RGB-D	Red Green Blue Depth	2
SLAM	Simultaneous Localisation and Mapping	1

IMU	Inertial Measuring Unit	3
RL	Reinforcement Learing	5
ANN	Artificial Neural Network	6
GPS	Global Positioning System	6

Chapter 1

Introduction

1.1 Background

There are many applications where vehicles are required to traverse rough terrain, such as in mines, rescue operations, agriculture, construction, etc. In many of these use cases rough terrain makes the use of wheeled, or even tracked, vehicles difficult or impractical.

Compared to wheeled robots, legged robots could perform better in many of these environments, allowing navigation over terrain that would be impossible for wheeled or tracked vehicles to navigate. While legged robots possess extreme degrees of potential terrain traversability, advanced control and sensory systems are required to realise this potential.

1.2 Research Goal

The overarching goal of this project is to design and implement a sensory and control system that will allow a hexapod robot to autonomously walk over rough terrain.

This goal of the project is broken up into the following sub objectives:

1. Obtain a mathematical model of the robot, its actuators and its sensors.
2. Create a model of the robot in a simulation environment for development and testing.
3. Implement a vision based Simultaneous Localisation and Mapping (SLAM) system.
4. Develop a real time vision based dense mapping system for use in anchor point selection.

5. Develop a optimisation system to select optimal end effector anchor points based on the surrounding terrain.
6. Implement tilt stabilisation feedback control.
7. Implement and test the entire system on the physical hexapod robot.

1.3 Methodology

When deciding how to determine optimal end effector placement various sensing methods were considered, such as using a Red Green Blue Depth (RGB-D) camera to view the environment, placing force sensors on the robots end effectors or measuring servo torque to determine when the end effectors were in contact with a surface. A previous paper by Erasmus *et al.* (2023) used a RGB-D camera by storing past snapshots to adjust the end effectors to the optimal height, it was decided that the primary sensing method for this thesis would also be a RGB-D camera but instead of storing snapshots, a height map would be generated of the local environment. This would allow for more advanced methods of anchor point selection.

The first step in realising this system was to construct a accurate simulation of the hexapod. The primary simulation packages that were considered are Gazebo, PyBullet and Multi-Joint dynamics with Contact (MuJoCo). Gazebo was a appealing choice due to the easy integration with ROS, however it was decided to use MuJoCo since it was found to have superior contact physics simulation (Erez *et al.*, 2015).

Once the hexapod was adequately modelled in MuJoCo a tripod gait state machine, IK system and control interface was implement, at this stage the hexapod was capable of walking on flat terrain.

Next the the system to generate the height map was implemented, this entailed sampling the RGB-D camera and comparing cells in the height map against the depth buffer. Once the height map was implemented it was possible to build the system responsible for end effector placement, this is covered in detail in chapter 6, after which collision checking for the generated end effector motion was implement, ensuring that the hexapod does not get stuck on pieces of terrain.

With this the system was realised in simulation, next the system was implemented and tested on the physical robot, discussed in detail in chapter 7

1.4 Scope and Limitations

As the hardware used was developed by Erasmus *et al.* (2023) this project will focus only on developing the necessary software to control het robot hardware.

The velocity control, tilt angle stabilisation and end effector motion planner was developed by the author, while the low level IK controller used was developed by (Erasmus *et al.*, 2023). The scope of this project does not include autonomous waypoint navigation and thus requires a human operator to provide desired velocity commands. If no solution can be found for the given velocity command the system will not attempt to adjust the velocity command, the human operator will be required to adjust the command.

The local dense height map system was developed by the author, while the SLAM system used, ORB-SLAM3 was developed by Campos *et al.* (2021). It should be noted however that ORB-SLAM3 does generate a global sparse feature map of the environment, thus implementing waypoint navigation should be a trivial addition.

The sensors used in this project are limited to a single RGB-D camera, thus even with the generation of a local map, there could be cases where the system will not have height data around a desired anchor point. No torque or touch sensors are used to augment the system, thus if a leg were to collide with the terrain the robot will not adjust its trajectory. The system will however attempt to choose a step path based on the local heigh tmap such that no collision occurs. Additionally no Inertial Measuring Unit (IMU) is used, pose estimation is entirely handled by the SLAM system.

1.5 Thesis Outline

Chapter 2 provides a literature review on the methods of control, sensing and simulation used for hexapod robots.

Chapter 3 provides a overview of the hexapod hardware and the modelling thereof. This includes the robots mechanical form, sensors, on board computers and the simulation environment that is used.

Chapter 4 describes the environment mapping systems used, this includes the local dense height map and the sparse SLAM system.

Chapter 5 covers motion related topics, this includes the walking gait, IK and effector motion planning.

Chapter 6 describes the optimisation function and its various scores used to acquire the optimal end effector anchor points during each step taken.

Chapter 7 covers the hardware implementation process and software structure on the hardware.

Chapter 8 describes the various tests preformed and results obtained thereof.

Chapter 9 provides the conclusion of the research and any recommended future additions.

Chapter 2

Literature review

This chapter provides an overview of past research done regarding the control of hexapod movement and sensing methods. First a brief history of hexapods is presented after which various terrain sensing and adaptation methods are presented.

2.1 Hexapod history

Hexapoda, Greek for "six legs" refers the group of arthropods possessing three pairs of legs. As an example see a flesh-fly in Figure 2.1.

Figure 2.1: A Flesh-fly

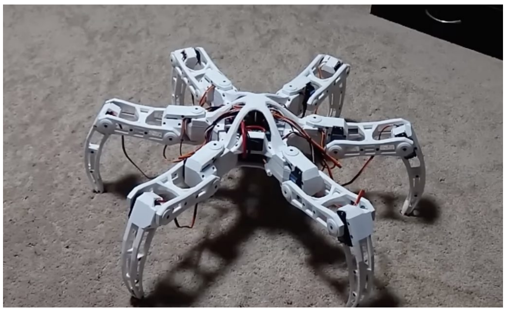

Figure 2.2: A circular hexapod

In the context of robotics "Hexapod" is used to refer to any robot with six legs, the most common configuration of Hexapods are either a rectangular layout with three legs on either side mimicking biological Hexapoda, or a circular design with radially symmetrical leg spacing, as seen in Figure 2.2

The hexapod possess the minimum number of legs to allow a naturally stable platform since while taking a step there can be upwards of three anchor points around the center of mass at all times. This makes the hexapod hexapods an ideal platform to navigate complex terrain while maintain stability, without requiring advanced balancing control systems.

For a hexapod to walk it must lift some of its legs while bracing with others, the number of swinging to bracing legs, and how each is moved, is referred to as the walking "gait". The chosen gait influences the speed and stability of the hexapod, the tripod gait is considered to be the most well rounded, having good speed and stability. In the tripod gait three legs are bracing while the remaining three swing. A example of a more stable gait would be the One by One gait, where only one leg is moved at a time.

It is also possible to create a system where there is no predetermined gait, but rather the system determines the optimal legs to brace and swing depending on the current walking environment.

2.2 Control

Walking over rough terrain requires a control system to correctly actuate the hexapods legs. Various types of control schemes exist, the primary schemes are traditional controllers, bio-inspired controllers and Reinforcement Learing (RL). These three schemes are discussed below. Control trends can be seen in Figure 2.3

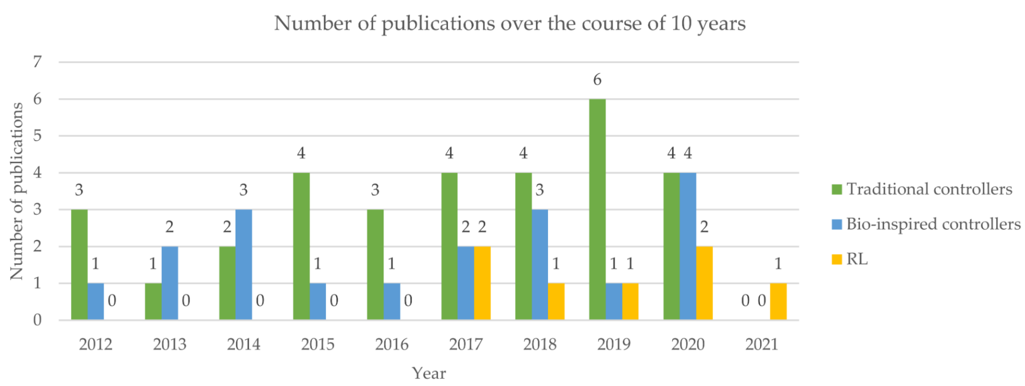

Figure 2.3: Trends of hexapod control schemes (Coelho *et al.*, 2021)

2.2.1 Traditional

Traditional controllers rely on an exact mathematical model of the robot and IK to calculate angular commands for all leg joints. This method of control is purely kinematic does not take into account external forces applied to the robot, thus it does not inherently adjust to the environment.

Instead of a purley kinematic model, a dynamic model can also be used. Using a dynamic model the forces acting on the robots legs are taken into account usually acquired through torque measurements from servos. By taking applied torque into account dynamic model controllers will intrinsically detect a deviation when an external force is applied to the robot or its legs and compensate appropriately.

It should be noted that it is possible for a kinematic model controller to also adjust to external disturbances, but this is not intrinsic to the control model and requires additional control logic.

2.2.2 Bio Inspired

Bio inspired controllers attempt to mimic the neural structure of animals to achieve the same locomotion methods that they use. This is implemented through the use of a Artificial Neural Network (ANN) If implemented successfully a bio inspired controller can be highly adaptable to the surrounding environment and is even able to adapt to damaged or missing legs.

2.2.3 Reinforcement Learing

RL controllers are created through using trial and error to construct a neural net that minimises a cost function for a specific goal. This theoretically allows RL controllers to adapt to any circumstances given enough time, allowing a very hight level of autonomy, as no prior direction is required. RL controllers are though notoriously difficult to train properly, especially when the amount of sensors and control outputs grow large, increasing the feature space. And event he most well trained RL agent still has the possibility to exhibit inexplicable behaviour.

2.3 Sensing Methods

No matter the control scheme used, to know where to place its feet the robot requires sensor(s) to sense its environment in some way, this could be achieved through simple sensors such as servo torque or touch. More advanced methods such as vision or Light Detection and Ranging (LiDAR) are also used. Homberger *et al.* (2017) use stereoscopic vision to adjust end effector height and to classify surface materials.

Depending on the terrain navigation system it might be required to localise the robot in 3D space, for this it is possible to use external sensors such as a type of beacon (RF, Reflective, Ultrasonic), Global Positioning System (GPS) or, through the use of a SLAM system, internal sensors such as vision could be used.

2.4 Simulation Environment

The most popular physics simulators for robotics in recent times are Gazebo, MuJoCo and CoppeliaSim (previously V-REP) (Collins *et al.*, 2021). Gazebo and CoppeliaSim both have easy to use Graphical User Interface (GUI) interfaces and easy integration with Robot Operating System (ROS). MuJoCo on the other hand does not have a full GUI interface, only a simulation viewer, and does not have native ROS integration. Having said this MuJoCo was found to be the most accurate and fastest simulator when considering the use case of robotics (Erez *et al.*, 2015).

2.5 Research Decisions

This research paper focuses on using vision based mapping to select the optimal anchor points for end effectors, thus it was decided that traditional kinematic control is used as it is the simplest and most predictable method of control.

The camera that is used is the Intel Realsense D435i RGB-D camera, incorporating stereoscopic cameras and a LiDAR sensor. This camera also has a extensive existing codebase and support that ensures accurate depth readings are attained.

A system to localise the robot within its environment is required, as the primary sensor used is an RGB-D camera, various visual SLAM systems were considered. ORB-SLAM 3, a optimisation-based, sparse map SLAM system was chosen to be used. ORB-SLAM 3 maintains a sparse map, an atlas, of both active and dormant features. This atlas is used to localise in the sparse map (Macario Barros *et al.*, 2022). The implementation of a dense map to be used for end effector placement is discussed in chapter 4. Compare this more to others?

Finally a simulation environment was chosen. Considering that the only relevant downside to MuJoCo is the lack of native ROS integration and the lack of a comprehensive GUI, which seeing as MuJoCo has good python bindings, could be seen as a advantage, MuJoCo was chosen as the simulator.

Chapter 3

Modelling

This chapter covers the simulation environment and modelling of the hexapod in MuJoCo.

3.1 Hexapod Construction

3.2 Servo Modelling

3.3 Simulation Terrain

Chapter 4

Mapping

In this chapter the localisation and mapping systems will be covered.

4.1 Localisation and sparse map

4.2 Dense map

Chapter 5

Motion

This chapter describes the systems governing the motion of the robot, such as leg motion planning and gait generation.

5.1 Gait State Machine

5.2 Kinematics

When commanding a foot position, the servo controller requires a function to calculate servo angles. While the foot arc planner, see section 5.3, requires the current position of the feet to function. The IK and FK functions described in this section provide this functionality. Figure 5.1 illustrates the leg geometry and variables used in the IK and FK functions.

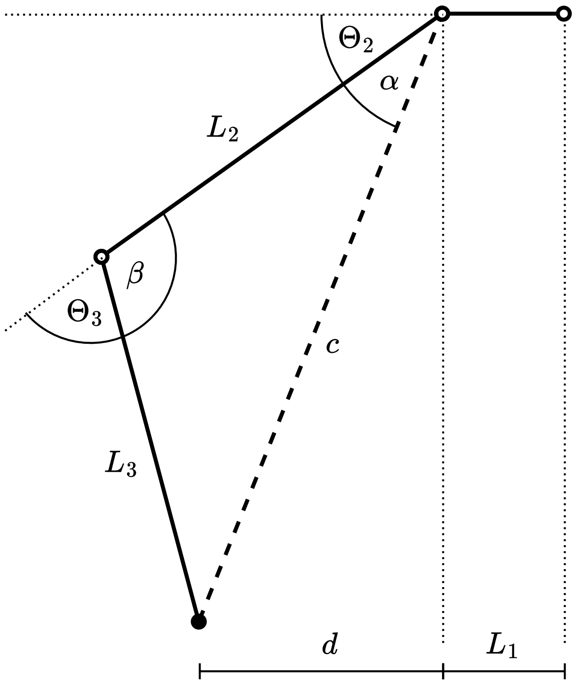

Figure 5.1: Leg Kinematics Diagram

5.2.1 Inverse Kinematics (IK)

The IK function calculates the leg servo angles, $\boldsymbol{\Theta} = [\Theta_1, \Theta_2, \Theta_3]^T_,$ required to move the foot to the given target position vector, $\boldsymbol{p_t} = [x_t, y_t, z_t]^T_.$ Equation 5.1 describes the IK function.

$$\boldsymbol{\Theta} = \begin{bmatrix} \Theta_1 \\ \Theta_2 \\ \Theta_3 \end{bmatrix} = \begin{bmatrix} \arctan\left(\dfrac{x_t}{y_t}\right) \\ \dfrac{\pi}{4} - \alpha - \arctan\left(\dfrac{y}{d - L_1}\right) \\ \dfrac{\pi}{2} - \beta \end{bmatrix} \tag{5.1}$$

Where α, β, c and d are calculate as shown in equations 5.2 to 5.5.

$$\alpha = \arcsin\left(\frac{L_3 \sin\beta}{c}\right) \tag{5.2}$$

$$\beta = \arccos\left(\frac{L_1^2 + L_2^2 - c^2}{2L_1L_2}\right) \tag{5.3}$$

$$c = \sqrt{(d - L_1)^2 + z_t^2} \tag{5.4}$$

$$d = \sqrt{x_t^2 + y_t^2} \tag{5.5}$$

5.2.2 Forward Kinematics (FK)

The FK function calculates the position vector of a foot, $\boldsymbol{p_c} = [x_c, y_c, z_c]$, given the current angles of the leg servos, $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]$.

$$\boldsymbol{p_c} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} d\cos\theta_1 \\ d\sin\theta_1 \\ L_2\sin\theta_2 + L_3\sin(\theta_2 + \theta_3) \end{bmatrix} \tag{5.6}$$

Where d is calculated as shown in in equation 5.7.

$$d = L1 + L_2\sin\theta_2 + L_3\sin(\theta_2 + \theta_3) \tag{5.7}$$

5.3 Foot Arc Generation

When taking a step the foot can not simply be moved to its destination in a straight line, as doing so will cause the foot to be dragged on the terrain, impeding the movement of the robot. Thus it is required to move the foot in an arc like motion to clear any obstacles that might be in its path.

5.3.1 Existing System

The existing system will at the start of each step compute an arc at the start of each step the robot takes, this arc is then sent to the servo controller to be executed. While efficient and effective in ideal conditions, this method of defining the arc has poor performance when considering external influences. If for example the robot has to adjust the final target of its feet mid step, this arc would have to be recomputed in its entirety, thus leading to possible performance concerns.

In addition to this Text, the current system is designed with the assumption that the starting position of the foot is grounded, thus if the arc is recomputed mid step the arc will be undesirable, as it will rise with the desired step height for a second time. This is illustrated in Figure 5.2.

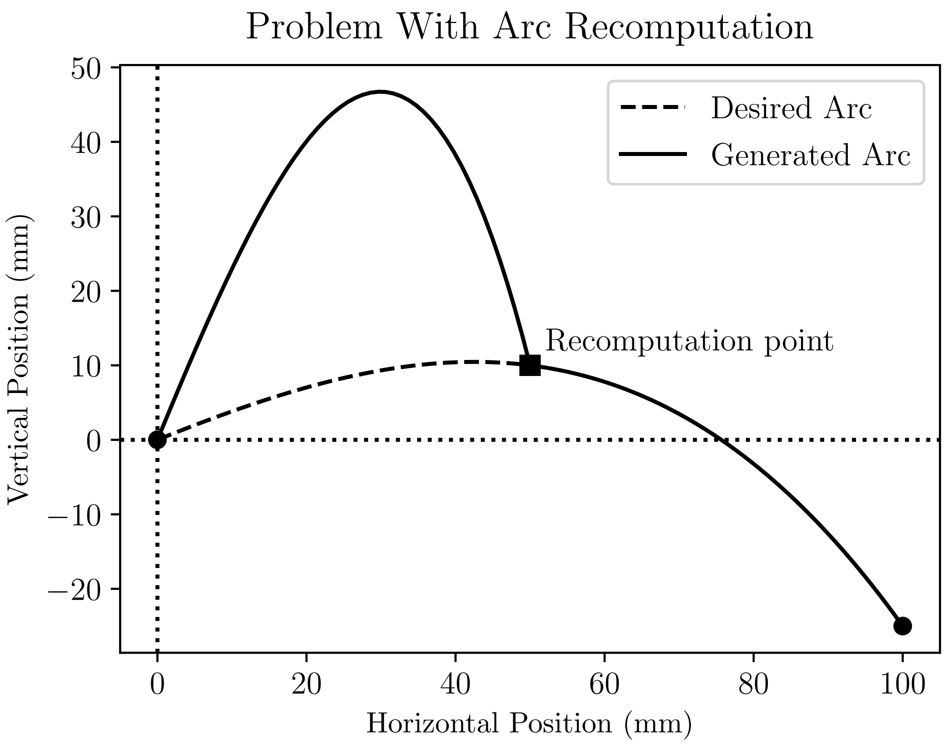

Figure 5.2: Existing arc recomputation problem

5.3.2 Improved System

The improved system solves this problem by utilising a flow function. During a step, this function will continuously calculate the direction that the foot must move to reach its destination. Thus this system is resilient to external disturbances and is capable of adjusting to varying destination and step height requirements.

The flow field is designed to first move the foot vertically upwards until horizontal coplanar with the destination, and then to follow a arc to the destination with a defined step height, this can be adjusted to make the arc start before or after coplanar. The step height can be adjusted at any point in time and the flow field will adjust accordingly. Figure 5.3 illustrates the field function and is described in section 5.3.2.1.

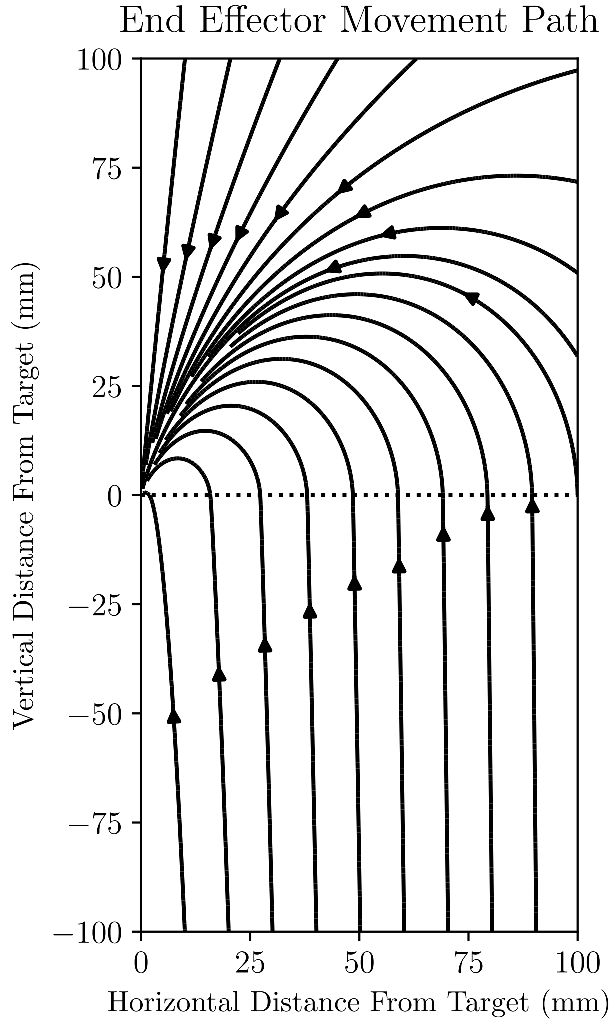

Figure 5.3: End effector movement path

5.3.2.1 Flow Function Description

The flow function, $\omega(x, y)$, uses the gradient function of a parabola passing through the point $[0, 0]$ and $[x, y]$ as a basis, where point $[x, y]$ is the current point that is being evaluated and x is the horizontal distance between the destination and the current point and y the vertical distance. The final function is described by equations 5.8 to 5.11, for the process of designing the flow function please see appendix **??**.

$$\begin{aligned} 2\omega(x, y) &= \frac{\delta}{\delta x \delta y} \quad f_a(x, y)x^2 + f_b(x, y)x + C \\ &= \quad 2f'_a(x, y) + f'_b(x, y) \end{aligned} \tag{5.8}$$

Where $f'_a(x, y)$ and $f'_b(x, y)$ are defined as follows:

$$f'_a(x, y) = -\left|\frac{v_h}{x}\right| - |S(y)| \tag{5.9}$$

$$f'_b(x, y) = \frac{y}{x} - f_a(x, y) \tag{5.10}$$

Where v_h is the variable describing the step height and $S(y)$ is a sigmoid like function defined in Equation 5.11.

$$S(y) = \frac{0.6(y - q)}{1 + |y - q| - 0.59} \tag{5.11}$$

Where q is the variable that determines at which vertical displacement the leg path transitions from primarily an vertical motion to an arc motion.

Chapter 6

End Effector Anchor Point Adjustment

The best possible anchor points for the supporting feet, given an initial point from the walking gait state machine, must be found based on the heightmap. This is done by applying various scores to the heightmap and selecting a point with the best score within an allowable deviation from the initial point received from the gait state machine.

6.1 Scoring

The scores considered are the gradient and proximity scores. The gradient score aims to reject points with high gradients while the proximity score rejects points close to other parts of the terrain with steep inclines, i.e. to reject points inside holes or close to ledges.

6.1.1 Gradient Score

The gradient score is simply taken as the slope of the terrain at the current point. The aim of this score is to prevent the robot from slipping due to selecting anchor points with too steep of a gradient.

As the heightmap slope is not defined by a known function, the gradient is calculated using the Sobel operator (Sobel (2014)), a combination of a central finite difference and a smoothing operator.

Equation 6.1 and 6.2 describe the two separable x and y direction kernels, G_x and G_y. These kernels are a combination of central finite difference and smoothing operator, see Appendix **??** for a breakdown. Equation 6.3 combines G_x and G_y to produce the gradient score, S_g.

$$G_x(x,y) = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \boldsymbol{h}_{i,j} \tag{6.1}$$

$$G_y(x,y) = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \boldsymbol{h}_{i,j} \tag{6.2}$$

$$S_g(x,y) = C_g\sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{6.3}$$

Where C_g is the weighting constant associated with the gradient score, S_g. The $*$ operator indicates the dot product.

6.1.2 Terrain Proximity Score

The proximity score aims to bias the selected anchor point away from points near steep inclines in the terrain. This score is defined as the average of the height difference of the current point weighted by the gaussian kernel $\boldsymbol{K}$, of size n by n. The distance around inclines that is rejected depends on the the chosen size of the kernel $\boldsymbol{K}$, the standard deviation of $\boldsymbol{K}$ and the height differences. This score is described in Equation 6.4.

$$S_p(x,y) = C_p\left|\frac{\sum[\boldsymbol{K} * (\boldsymbol{h}_{i,j} - \boldsymbol{h}_{x,y})]}{n^2}\right| \tag{6.4}$$

$$x - \tfrac{1}{2}(n-1) < i < x + \tfrac{1}{2}(n-1) \text{ and } y - \tfrac{1}{2}(n-1) < j < y + \tfrac{1}{2}(n-1)$$

Where x and y are the indices of the cell whose score is currently being evaluated, $\boldsymbol{h}$ is the heightmap and C_p is the score's weighting constant. The $*$ operator indicates the dot product.

A diagrammatic, sliced, representation of the proximity score can be seen in Figure **??**. The slice is taken along the x axis of the heightmap.

6.1.3 Constraints

It is important to constrain the possible anchor points to confer to the stability triangle, meaning that the centre of mass of the robot must be inside the triangle formed by the three anchor points. Additionally, it is important that the points selected are not too far away, both in the horizontal and vertical direction for the robot to reach.

6.2 Method of adjustment

Once the heightmap has been processed into the score map, which is done by adding the slope score and terrain proximity score, the point with the best score must be found for every initial anchor point. The resolution of the heightmap is not very high and the adjusted anchor point is not allowed to deviate too far from the initial anchor point, additionally due to the parallel nature of the heightmap generation it is possible to score the entire heightmap with minimal cost. Thus, it was decided to not employ an optimisation algorithm, such as gradient decent or Bayesian, but rather to use a radial search algorithm. This algorithm progressively expands its searching radius over the square score grid until a valid score is found, at which point it terminates, thus ensuring that the closest valid point to the initial anchor point is selected. See Figure **??** for a diagram representing the search pattern for a 5 grid squares search area. Note that this search pattern will become inaccurate for large search areas, as the pattern steps in a square manner. This however is not much of a concern for smaller search areas.

Chapter 7

Hardware Implementation

This chapter describes the process of implementing the system built in previous chapters on the physical robot.

Chapter 8

Testing

This chapter covers all tests performed to validate performance of the system.

Chapter 9

Conclusions

Appendix A

Mathematical proofs

A.1 Euler's equation

Euler's equation gives the relationship between the trigonometric functions and the complex exponential function.

$$e^{i\theta} = \cos\theta + i\sin\theta \tag{A.1}$$

Inserting $\theta = \pi$ in (A.1) results in Euler's identity

$$e^{i\pi} + 1 = 0 \tag{A.2}$$

A.2 Navier Stokes equation

The Navier–Stokes equations mathematically express momentum balance and conservation of mass for Newtonian fluids. Navier-Stokes equations using tensor notation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\left[\rho u_j\right] = 0 \tag{A.3a}$$

$$\frac{\partial}{\partial t}\left(\rho u_i\right) + \frac{\partial}{\partial x_j}\left[\rho u_i u_j + p\delta_{ij} - \tau_{ji}\right] = 0, \quad i = 1, 2, 3 \tag{A.3b}$$

$$\frac{\partial}{\partial t}\left(\rho e_0\right) + \frac{\partial}{\partial x_j}\left[\rho u_j e_0 + u_j p + q_j - u_i \tau_{ij}\right] = 0 \tag{A.3c}$$

Appendix B

Experimental results

List of references

Campos, C., Elvira, R., Rodríguez, J.J.G., Montiel, J.M. and Tardós, J.D. (2021). Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890.

Coelho, J., Ribeiro, F., Dias, B., Lopes, G. and Flores, P. (2021). Trends in the control of hexapod robots: a survey. *Robotics*, vol. 10, no. 3, p. 100.

Collins, J., Chand, S., Vanderkop, A. and Howard, D. (2021). A review of physics simulators for robotic applications. *IEEE Access*, vol. 9, pp. 51416–51431.

Erasmus, S. *et al.* (2023). Guidance, control, and motion planning for a hexapod robot moving over uneven terrain.

Erez, T., Tassa, Y. and Todorov, E. (2015). Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4397–4404.

Homberger, T., Bjelonic, M., Kottege, N. and Borges, P.V. (2017). Terrain-dependant control of hexapod robots using vision. In: *2016 International Symposium on Experimental Robotics*, pp. 92–102. Springer.

Macario Barros, A., Michel, M., Moline, Y., Corre, G. and Carrel, F. (2022). A comprehensive survey of visual slam algorithms. *Robotics*, vol. 11, no. 1, p. 24.

Sobel, I. (2014 02). An isotropic 3x3 image gradient operator. *Presentation at Stanford A.I. Project 1968.*