Predicting Gaits Based on Prior Experience

Using the Monte Carlo Method

Hans Fredrik Fahle

Thesis submitted for the degree of
Master in Robotics and Intelligent Systems

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2017

Predicting Gaits Based on Prior Experience

Using the Monte Carlo Method

Hans Fredrik Fahle

Predicting Gaits Based on Prior Experience

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Robots are becoming a valuable part of the society. An important reason behind this is that the robots are becoming more capable of operating without human interactions. Today the robots are capable of moving around with certain limitations. This gives the opportunity to increase the value that robots have by improving their ability to move around independently.

This study aims to determine if prediction, utilizing prior experience, can be used to find the most suitable gait for traversing a certain type of terrain. The robot is a four-legged robot with five different gait configurations. The accumulation of the experience data is done by making the robot traverse five different terrain types using the five different gaits. For this process the Monte Carlo method was used. The experience data accumulated is then used to predict the most suitable gait when traversing each terrain type on a test terrain.

To evaluate the performance of the prediction approach a second approach is implemented. The second approach is a traditional approach which evaluates the stability of the robot while traversing a test terrain.

The final result in this thesis compare the prediction approach with the traditional approach. This comparison gives an indication of how well the prediction approach performs.

Contents

List of Figures

List of Tables

Acknowledgements

The writing of this thesis has come together with the help of many people. I would like to thank you all for the kind help that you have all given me.

I would like to thank all my fellow students and professors at the Robotics and Intelligent Systems research group for making a great learning environment.

Kyrre Glette and Tønnes Nygaard, it has been good to have you as advisors and a source of knowledge; I greatly appreciate you. A special thanks to Tønnes Nygaard for letting me use your robot in my experiments, and for all the help during implementation.

Finally, I will thank my family and friends. Thank you Erica, my English would not have been the same without you. I would like to offer special thanks to my sister, Ine, for being a discussion partner and for offering valuable feedback throughout this work.

Chapter 1

Introduction

1.1 Motivation

Until recent years robots have only been used in industry to do repetitive and simple tasks. In recent years there has been much research in the field of robotics and machine learning, which allows the robots to learn from performing different type of tasks and adjust accurately to changes in data. These improvements gives the robots the opportunity to operate without help of human interactions, meaning that they can make decisions by themselves and make predictions based on the gained experience from the learning process. The progress in the field of robotics has opened for a large amount of different opportunities.

In the past years, many types of moving robots have been built and experimented with. The first moving robots built were robots that were either programmed to walk forward, do a certain task, or operated by a human. These moving robots also varied in functionality and design, and the most common configurations used wheels or legs. In recent years, there have been many attempts to make the robots capable of moving around independently. These attempts have ended in both failure and success. The most known example is the BigDog[1].

The main challenge with moving robots is traversing different types of terrain. A wheeled robot has no difficulty traversing flat terrain, but as soon as the terrain gets rough a wheeled robot has great difficulty traversing the terrain. With a legged robot, there is more flexibility: a legged robot can change and use different gait configurations in order to successfully traverse rough terrain. It is interesting to explore how a legged robot will handle different terrain types.

Recent research in gait optimization has largely focused on collecting data about the present and then adjusting the robot by selecting the most optimal gait. With machine learning it is possible to learn how different gaits work on different types of terrain. With a learning process a robot can accumulate and use knowledge about how different gaits will work on different terrain types. Having this prior knowledge, the robot would be able to make a prediction and decide which gait thdt would be the most suitable for the terrain before even starting to traverse the terrain.

1.2 Goals and Research Questions

The main goal of this thesis is to determine whether or not prediction can be used to find the most suitable gait for a given terrain based on past experience. A suitable gait in this scenario is the fastest gait that traverse the terrain. To present the main goal a problem definition has been formalized (1.1).

(1.1) Is it possible to use prediction to find the most suitable gait for a certain terrain type using prior experience'?

In order to explore the problem definition (1.1), two research questions have been defined and will be discussed in this thesis.

To be able to make a prediction a machine will need to gain experience. In this thesis a robot would need to gain experience using the different gaits on different terrain types. There are many different methods of machine learning. One of the sub goals of this thesis is to use and evaluate the Monte Carlo method to gain experience. This sub goal is formalized in research question (1.1).

(1.1) Is the Monte Carlo method a viable method to gain experience about the performance of each gait?

In the past, many experiments have successfully found the most suitable gait for a certain terrain type. Some of these approaches are also commonly used to find the most suitable gait. The other sub goal in this thesis is to evaluate the performance of the prediction approach, used in this thesis, by comparing it with a traditional approach. This sub goal is formalized in research question (1.2)

(1.2) Will a prediction approach perform better than a traditional approach?

To answer these questions the use of a robot with five gait configurations is used. The robot gains experience by traversing five different terrain types using the five gaits. Finally, the robot will use the gained experience to predict the gait that would get the robot from one point to another on a test terrain the fastest way. The results will then be compared to a traditional approach.

1.3 Outline

The thesis is divided into six chapters: introduction, background, implementation, experiments and result, discussion, and conclusion and future work. Chapter 2, Background, presents the previous work related to the thesis and explains the different theories use in the implementation. Chapter 3, Implementation, presents the implementation environment and how both the prediction and traditional approaches are implemented. Chapter 4, Experiments and results, explains the experiments used to accumulate the data, presents the results, and evaluates the performance of both the prediction and traditional approaches. Chapter 5, Discussion, is a discussion around the implementation and results obtained in this thesis. Chapter 6, Conclusion and future work, attempts to draw a conclusion from the results obtained and presents the possibilities regarding the implementation used in this thesis.

Chapter 2

Background

2.1 Learning

To be able to make a prediction it is necessary learn and gain experience. This section will present the theory behind the process of learning and how prediction works.

2.1.1 Internal models

For as long as animals has lived they have had the ability to learn from performing different tasks. After preforming a certain task the animal stores the experience from the learning process. With all their past experiences and current sensing, they are able to make an internal model of the environment and their own body structure. When the animal perform the same task again it could use the internal model to improve its performance by predict the most suitable action. The animal will have the ability to adapt to sudden changes in the environment and their body, based on past experiences. This means that the internal model has an effect on animals judgment and actions performed.

The existence of the internal models in animals was first discovered in the Helmholtz experiment[2]. He did an experiment which proved that rather than sensing the position of the eye, the position is predicted by the motor command that works on the eye. These models, that we know exist in animals, can be transferred to robotics. This means that robots can use an internal model to calculate the most suitable action. These internal models would incorporate the world model, which is the environment surrounding the robot, and the robot model, which would be the model of the robot structure. The world model contains information about objects in the environment. These objects can be static obstacles, hazards or dynamic obstacles. Dynamic obstacles can be a moving obstacle, or a human that the robot needs to interact with. With the information from the internal model, consisting of both the world model and robot model, the robot could predict the most suitable action given a certain situation. To predict the outcome of a certain situation, the robot not only has to calculate the next action, but find the appropriate action in the specific situations. With the internal model the robot has the ability to do an internal simulation, which means that the robot can find the most suitable action in different situations, for example, if the robot got damaged. The problem with most methods used for the internal model is that they do not have the opportunity to adapt if the morphology of the robot suddenly changes. To be able to predict, a robot needs to be capable of developing more than just one internal model, so it can select the model that is the best fit after a sudden change.

2.1.2 Machine learning

As long as computers and machines have existed they have been instructed on what they need to accomplish. In resent years however, machine learning has had a big influence on the way computers and machines are being developed. Machine learning makes the machine capable of learning and using prior knowledge to make decisions independently, without the help of human interactions.

Definition

Machine learning is similar to the process of data mining[3], where the object is to look for patterns in data. The difference between machine learning and data mining is that machine learning uses data to learn and adjust accordingly to the change in input data. The basic concept of machine learning and how it is used to predict unseen data is illustrated in figure 2.1.

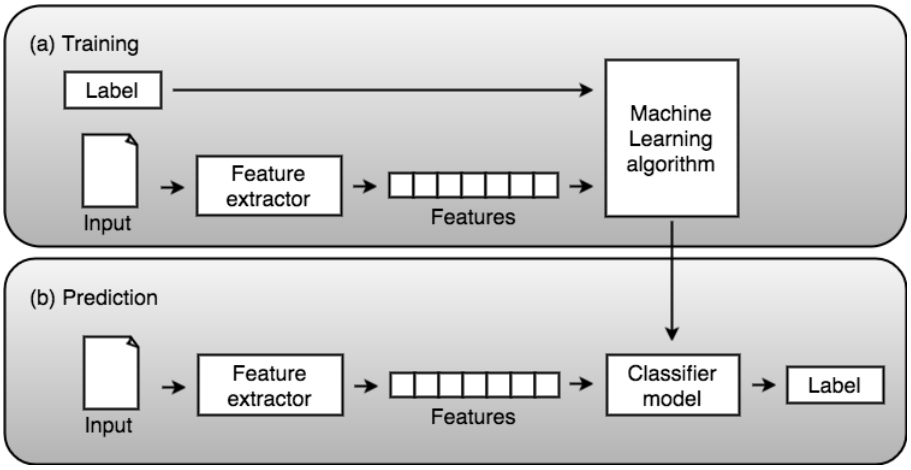


Figure 2.1: The basic concept of machine learning consist of a machine learning algorithm/classifier with labeled features as input, shown in (a). Then algorithm produces a classifier model to predict a unseen/new feature, shown in (b)[4].

Sample data

As shown in figure 2.1 sample data is required in order to learn. Sample data is presented to the machines as data sets. There should be a correlation between the sample data and the desired outcome. In order for the machine learning algorithm to generate a durable model, the sample data would need to suit the problem. There are a few complications with sample data, which can make it hard to generate a durable prediction model. The complication might be that the data could change over time or that there is a high degree of sample noise. These complications should be avoided in order to collect a good set of sample data. The sample data is divided into three sets, training data, verification data and test data. The training

data set consist of a input and the desired output, and is used by the machine learning algorithm to generate the classifier model. The algorithm tries to find patterns and connections between the sample data. This process is repeated until the model has reached a minimal error threshold. The verification data set is then used to measure the performance of the machine learning algorithm. The test data set is a final test of the classifier model, and consist of data generated in the real world.

Supervised learning

The main objective of machine learning is to generate a classifier model based on a training data set, in order to make a prediction or decision. Each sample of the training data set comes in pairs consisting of input/feature and a corresponding desired output/label. The output/label is the target data of what the machine learning algorithm should end up with from the input. The target data can be generated by humans in order for the machine to learn. The training phase consists of analyzing each of the training pairs and ending up with a function that describes the separation between the classes. This function is referred to as the decision boundary, and is used to predict new or unseen data.

Supervised learning problems can be divided into two different sub groups, regression and classification.

Classification

The basic concept of a classification problem is to get a new or unseen input and then predict which class it belongs to. Before getting a new or unseen input, the different classes have been defined by the training data, which is labeled with desired output. Hence, a classification problem is a supervised learning algorithm. The classes get separated by the decision boundary, shown as the black line in figure 2.2.

Figure 2.2 is an example of a two-class classification problem. This problem is a non-linear problem and the function or decision boundary is produced from $f(x^3)$. The red circles and blue triangles are classified and belongs to the training set. The machine learning algorithm has generated the function for the classification problem, shown as the decision boundary. The grey square is the new/unseen feature, which the classifier will predict. Since it is on the right side of the decision boundary, the classifier is predicting it to be a part of the blue triangle class.

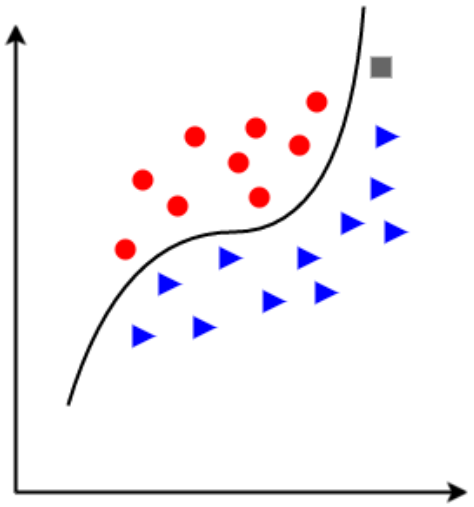


Figure 2.2: A two-class classification problem with a non-linear decision boundary, shown as the black line.

Overfitting and underfitting

Among the most common problem with classification problems are over- and underfitting, which occur because of the random noise/error in the training data set. Overfitting is when a classifier/algorithm has generated a model that defines the random noise/error in the training data set, instead of finding the true function that defines the separation between the classes. Overfitting occurs when a model contains more or too complex parameters. The decision boundary created when overfitted is overly complex and includes the random noise/error. A model that has been overfitted will have a poor prediction performance, since it will react to minor changes in the training data set. Underfitting is when a classifier/algorithm is not capable of finding the true function that defines the separation between the classes. Underfitting occurs when a model does not contain the necessary parameters to define the true function. A model that has been underfitted will have a poor prediction performance, since it will not be capable of classifying the new/unseen data to the correct class. In figure 2.3 is an illustration of a model which is underfitted, overfitted and optimal.

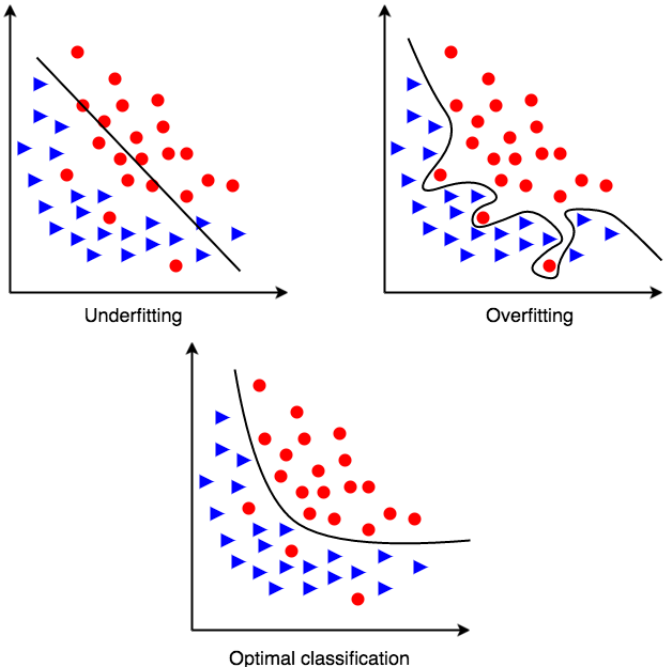


Figure 2.3: Illustration of an underfitted, overtfitted and optimal model

Benefits of Machine learning

The reason why machine learning has become somewhat commonly used is that it comes with a set of benefits. The first benefit is related to the big increase in amount of data and statistics[5, 6]. Machine learning enables machines or computers to analyze and classify data, which means that machines and computers are capable of operating and handling big amounts of data independently. To use machines to classify and predict by implementing machine learning has been used in many fields of study, for example in the financial sector[7], the academic sector[8], the medical sector[9], the industrial sector[10] and the photography sector[11]. Another benefit is that when a machine or computer can operate independently the process of analyzing data will be much more efficient, compared to a human analyzing the same amount of data.

2.1.3 The Monte Carlo method

The Monte Carlo method has been used in many different fields of study, for example, engineering[12], radar technology[13], financial markets[14], and computer science[15]. The Monte Carlo method is a group containing different computational algorithms, where the goal is to obtain numerical results using random sampling. The principle of the Monte Carlo method is to use randomness to solve a problem. Monte Carlo is used when the problem being solved is too difficult or impossible to solve using other approaches. One of the most commonly problem used to solve using Monte Carlo is optimization problems.

Basic concept The basic concept of the Monte Carlo is to generate lots of random samples and then do an observation on these samples[16]. Figure 2.4 is an illustration of the Monte Carlo method.

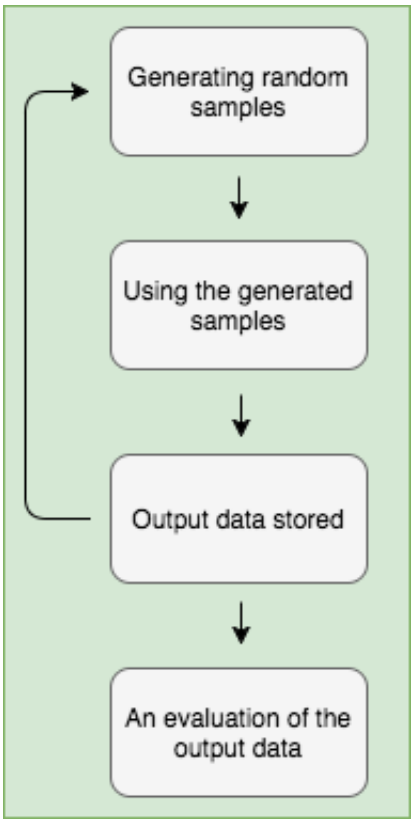


Figure 2.4: Illustration of the Monte Carlo method

The first step is to define the goal in order to find the solution to the problem, for example, the chance of getting the number three when rolling a dice. The first step is to generate random samples, where in the example, one random sample consist of rolling the dice once. Then the next step is to evaluate the random sample, which means evaluating a sample where the dice is rolled once. After these two steps are done, the result of the evaluation of that specific sample is stored. In order for the Monte Carlo method to work properly, these three steps are repeated: generating, evaluating and storing the random sample. This is an essential part of the Monte Carlo method. The Monte Carlo method repeats the three steps n number of times, where n is number of runs. The variable n is a static variable, which is set before running the Monte Carlo method. After running n number of random samples, the Monte Carlo method will end up with a solution to the problem, for example, the probability distribution of getting number three when rolling a dice.

The number of runs n is important to set to a suitable value. If the n is set too low it would not get a good representation of the random samples. If rolling the dice ten times it could either be very lucky and get seven threes

out ten or very unlucky and get two threes out of ten. Both scenarios are a bad representation of the probability distribution. The higher n, the closer the Monte Carlo method would get to the actual probability distribution, which is 1/6 for getting the number three when rolling a dice. However, the problem is that the n should not be set to high. This will be time-consuming and require more computational power. The effect of n is significant in order for the Monte Carlo method to perform well.

Benefits of the Monte Carlo method There are a few benefits to using the Monte Carlo method. The first benefit is that the Monte Carlo method easily generates a probability distribution, which can be used to predict the new/unseen data. A second benefit is that the Monte Carlo simulation often reaches the global optimum in optimization problems, if the n is set properly. A third benefit is that the Monte Carlo method can find solution to problems, which might be difficult or impossible to find a solution to for other approaches. A fourth benefit is that the Monte Carlo method does not require high-level mathematical understanding in order to be implemented. A fifth benefit with the Monte Carlo method is that it can be easily applied in different fields of study, which generates and uses different types of data.

2.1.4 Other machine learning approaches

The Monte Carlo method is not commonly used for optimization problems in robotics. This section will present a few approaches commonly used for optimization problems in robotics, and the disadvantages using these approaches.

Neural networks

Neural networks are a class of models that are inspired by the animal brain, consisting of interconnected computational elements (neurons). These neurons get input, processes the input, and give out an output. The neural network could also consist of a hidden layer, which is used to solve more complex problems. Figure 2.5 shows a simple neural network with one hidden layer.

A neural network consists of neurons which are connected, with weighted links, to other neurons. For a neural network to work, the neurons need to be trained, which means adjusting the weights to compute a reasonable output from the inputs. These neurons communicate with one another through links, which are weighted, in the network. After the neural network is trained, a machine or computer is able to use the network to predict. To make sure a neural network is a suitable fit for the robot, the performance of the network is tested over its entire lifetime. This test is summarized in a score, called fitness. To get this score, a robot receives a large amount of sensor data with various input. After all the data is passed through the network with the best fitness is used.

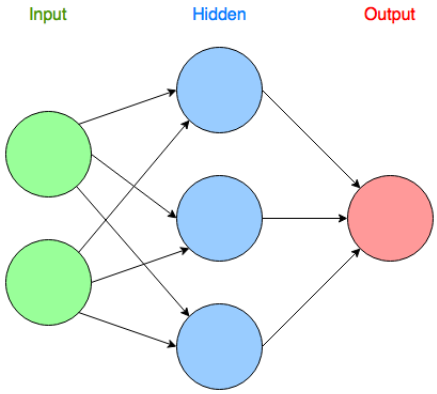


Figure 2.5: Neural network

Disadvantages of neural networks One problem with the use of neural networks is having a robust solution. What it means to have a robust solution is that the robot should handle sudden change in the environment or the structure of the system. When a system is exposed to a sudden change, the neural network needs to be trained again with the change in the system or environment. A basic neural network is a slow approach and there have been a few different attempts to update the weights during run time. Earlier attempts with neural networks has been Static neural networks, Plasticity neural networks and Neuromodulation which was performed by Norouzzadeh and Clune[17].

Evolution

Evolutionary algorithm is an optimization algorithm. Evolution has been experimented and modified over the years[18, 19, 20, 21].

Basic concept Evolutionary algorithm is an algorithm inspired by the processes from biological evolution. The processes used are biological evolution, such as reproduction, mutation, recombination, and selection.

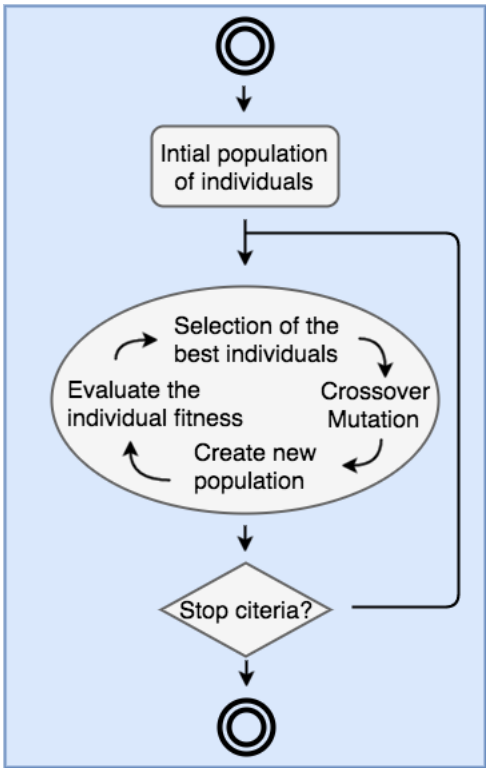


Figure 2.6: The general process of a biological evolution

The evolutionary algorithm begins with randomly generated population, where each individual in a population is a solution to the problem. If the problem is to find the fastest route home, one route home would be an individual in the population. Then the algorithm does an evaluation of the fitness of each individual in the population. After the fitness is evaluated, the individuals with the best fitness, referred to as parents in biology, are selected and used in the reproduction processes. New individuals or offsprings are then generated through the use of crossover and mutation. Then the new generation is created without the individuals with the lowest fitness. The process from evaluating each individual to the newly created generation is an iterative process and keeps repeating until the stop criteria is met. When the stop criteria is met the evolutionary algorithm has solved the problem.

Single- and multi-objective evolutionary algorithm The optimization problem solved by the evolutionary algorithms can be dived into two groups: single- and multi-objective optimization. In single-objective optimization the goal is only to optimize a single objective function. In multi-objective optimization, which involves having multiple objective functions, the goal is to optimize all the objective functions. Hence, the difference between single- and multi-objective optimization is the number of objective functions, which are simultaneously optimized. An objective function is a function representing the fitness, and the goal of the evolutionary algorithm is maximize the objective function. In multi-

objective evolution, all the individuals have multiple objective functions and in single-objective the individuals only have one objective function. If the optimization problem were to find the fastest route home, this would be a single-objective optimization problem, where each individual represents a route home with a respective fitness. If the optimization problem were to find the fastest route home with a stop at the grocery store, this would be a multi-objective optimization problem, where each individual represents a route to the grocery store with a respective fitness and a route home with a respective fitness.

Disadvantages with evolution There are a few disadvantages to using the evolutionary algorithms. One disadvantage is that evolutionary algorithms are not guaranteed to find the most optimal solutions in a finite amount of time. This depends on the complexity of the problem, and on very complex problems the time spent searching for the most optimal solution could be high or not finite. Another problem with evolutionary algorithms is that they might require significant computational power to find a solution. The reason is the loop from evaluating each individual to creating the new generation in certain problems might be complex.

Hill Climbing

The Hill Climbing algorithm is also an optimization algorithm. Hill Climbing has been experimented with, and modified, in the past.[22, 23, 24, 25]

Basic concept The algorithm starts with a random solution and searches for improvements to the solution in the search space. The search space is dependent on the problem the algorithm is trying to solve. If the problem is trying to find the fastest way home, the search space will consist of different routes to get home. If a new solution is found to be a more optimal solution, this solution will be replace with the solution the algorithm started with. The algorithm will repeat this process until it can not find further improvements to the solution. The reason why it is called the hill climbing algorithm is that the goal is to find a higher point(better solution) and it stops searching when it reaches the summit(the optimal solution).

Disadvantages with Hill Climbing The hill climbing algorithm is optimal to use on a convex problem, when there is only one local maximum or optimal solution. The problem with hill climbing is when the problem trying to be solved is not convex and unfortunately most of the real-world problems are not convex. When the problem is not convex, the Hill Climbing algorithm might only find the local maximum and not the global maximum. In order to find the global maximum, the random solution picked at the start would need to be the least optimal solution closest to the global maximum. In figure 2.7 is an illustration of the problem with the Hill

Climbing algorithm. The random solution picked at the start is marked as the red circle and the algorithm will only find better solutions towards the local maximum, where it will end up stopping.

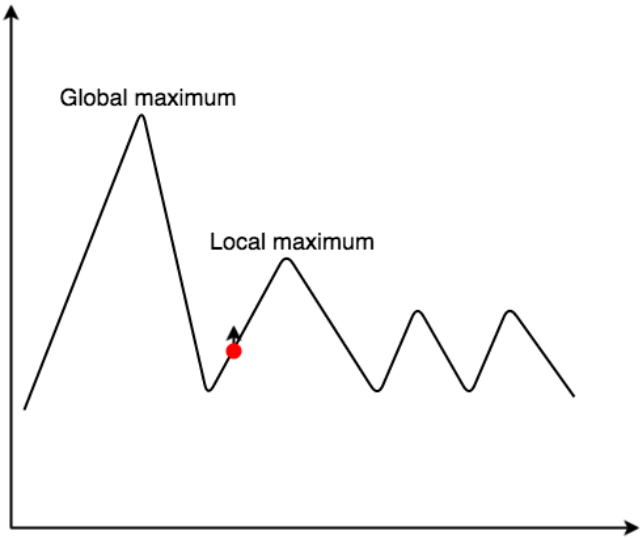


Figure 2.7: Illustration of the Hill Climbing algorithm. The "highest point", the most optimal/suitable solution, is called global maximum. The other peaks are local maximum, the most optimal/suitable solution in that area of the search space.

2.1.5 Prediction

The chapter has so far given a understanding of machine learning, this section will give an understanding of the prediction and how it is used.

Definition

Prediction The definition of prediction is the capability to make an estimate of an unknown event based on past experience or prior knowledge[26]. Having experienced that a rock is both heavy and hard, the next time a human mind will make a prediction that it would hurt if it fell on a toe, so the body tries to dodged it. With this definition in mind, we can see that there is a connection between internal models and predicting the next move. The capability to make a good prediction could make it easier to adapt to unexpected changes in the environment. There are also different types of prediction. You can make a prediction of what would happen in the future, or you can predict what is happening in the present with just looking at data. For example, you can predict that you will make the next basket when throwing a basketball, or you can predict that you are standing on grass when having your eyes closed. With predicting the future there is the possibility of making a decision before something happens. By being able to make this prediction, a human can can find the most suitable outcome of a situation before it happens. With predicting the present

it would be possible observe new and unexpected events. For example, if there is a constant noise a human brain will predict that it will continue until it suddenly stops and that will be experienced as unexpected. Regardless of predicting the future or the present, prediction can be beneficial in many different situations.

Use of prediction

Prediction has been used in various research in recent years. In the past years, predictions has been used to predict certain events. The ability to predict a certain event has become especially popular in order to predict the outcome of sports events[27, 28]. Being able to predict the outcome of difference scenarios has also been found to be useful, for example, in the financial sector. There has been much research focusing on the outcome of a stock on the stock market[29]. The use of prediction has also been applied in different fields of study, such as the medical sector[30] and engineering[31].

In robotics There has been some research in robotics where prediction is applied[32]. Robots used in the industry are only programmed to complete what they are intended to do and not able to anticipate certain scenarios. The ability to adapt and prepare for a new situation could be a significant benefit in certain scenarios. This type of prediction is found to be useful in robotics. When a robot is capable of predicting the most suitable action, is dependent on the situation, for example, it could avoid situations where it might get damage. An example is if an industry robot arm is moving towards an object blocking the why and the robot arm is capable of predicting the new position to avoid the object.

2.2 Legged robotics

So far this thesis has explained basic theory and concepts on learning and prediction. This section will first focus on how a legged robot can utilize basic concepts of forward movement. Then the section will give an understanding of using different types of gaits.

2.2.1 Introduction

In robotics there have been many attempts to get a robot to move independently, without help from external equipment. Most of the robots built are made for industry[33, 34, 35], which is stationary robots without the ability to move around. The first attempts to give a robot the ability to move around, where wheeled robots[36, 37]. In recent years, robots have been built with legs, which makes the robots more complex and gives a robot a greater ability to move around. The most known legged robot is BigDog[1].

In recent years, there has been much research in robotics using four-legged robots. Some of the robots that have been built are Littledog[38], cheeta-cub [39] and StarlETH [40]. These robots consist of a body and four legs attached. The four legs are responsible for the movement and stability of the robot. An example of a four-legged robot is shown in figure 2.8. This robot is being used for research at the University of Oslo. The research completed with four-legged robots has shown that they can handle uneven terrain[41] compared to robots with wheels[42]. The four-legged robots have the ability to walk over holes and bumps in the terrain. They have a morphology that makes them able to use a foothold to avoid obstacles or holes. Four-legged robots have the capability of changing gait to maintain stability, whereas the wheeled robots cannot change gaits. With the ability of changing gaits, the legged robots can then change gaits depending on the terrain type. However, even though legged robots can walk over uneven terrain, the challenge is to maintain stability[41]. Finding the most suitable gait is crucial in maintaining stability, and section 2.2.5 will explain how a robot can maintain stability. Having four legs makes it easier to maintain stability than, for example, having two legs. However, having more legs can make the task of finding a gait more complex[43]. There are more legs to control and the robot must make sure that all these legs are in the correct position.

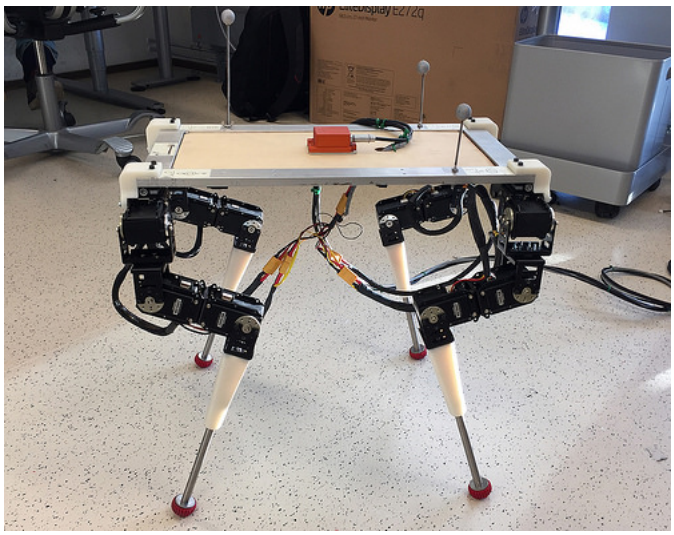


Figure 2.8: Four-legged robot developed at the University of Oslo

2.2.2 Terrain considerations

Type of terrains The natural environment on earth consist of many types of terrain. The terrain can vary from dry to wet, flat to rough and soft to hard. The different types of terrain in the environment can lay side by side and the terrain type can therefore suddenly change from dry to wet. It can also change during the day or year, which make the terrain sometimes very unpredictable. Another factor is the friction on the different types of

terrain. Friction can be high, which makes the robot stick to the surface, or the friction can be low, which can make the robot slip. Friction where rubber is in contact with rubber is high and makes it easy for the robot to move forward. However, where metal is in contact with wood the friction is low. Therefore, friction can make it difficult for a robot to move forward. Different types of terrain is shown in figure 2.9.

(a) Rocky terrain[1]

(b) Grassy terrain[2]

(c) Sandy terrain[3]

(d) Gravely terrain[4]

Figure 2.9: The images illustrate different types of terrain a robot can face.

All these factors make the environment and the different terrains unreliable, which could make a robot fall over. In order for the robot to handle variations in friction, it would need to get feedback that gives information about friction[44]. Therefore, robots will have to be able to adapt to these changes in the environment. The robots that are the most suitable for this task are four legged robots.

[1]https://cdnb.artstation.com/p/assets/images/images/005/195/365/20170310213105/small$_s$*quare*/*karen* − *stanley* − *rocks* − *a.jpg*?1489203066

[2]https://quickgrass.co.uk/wp-content/uploads/2015/05/stratford-2016-3.jpg

[3]https://www.sketchuptextureclub.com/public/texture$_d$/0019 − *beach* − *sand* − *texture* − *seamless* − *hr.jpg*

[4]http://www.bowlandstone.com/images/content/products/213$_9$05*apline* − *gravel*2*.jpg*

2.2.3 Gaits

The basic principle of locomotion in animals is that they lift one leg and move it to a new position. The coordination of lifting the legs and placing them in a new position is defined as a gait. Different gaits are characterized by the way the legs are lifted and moved. Lifting or placing the leg is known as an event of the gait and together they make up a specific gait.

Types of gaits In order to handle different types of terrains, the robot needs to be able use different gaits. A gait is how the legs are lifted and placed to a new position. For a robot with four different legs, the possibilities could almost be endless, but most of the gaits are not suitable for moving forward[45]. Animals change gaits due to the relationship between speed and being energy efficient. The reason why they change gait is that certain gaits are more suitable depending on the need for speed, stability and energy efficiency[46] on different types of terrain. By changing the gait of the legged robot, the robot can handle different types of terrain. When a robot walks over a flat surface it could be walking with long steps and a high center of gravity. However, when walking over rough terrain the robot should walk with a gait with small steps and a low center of gravity. Another factor which could affect the performance of a gait over a certain terrain is speed. By utilizing a gait with high speed, it might result in failure on rough terrain, but might be beneficial on flat terrain. Therefore, it is necessary to design a robot considering both the gait configuration and speed in order to handle different types of terrain.

Finding the most suitable gait

In the past, research has sought to find the optimal or suitable gait for a certain terrain. This section will how a robot controller is used to find the most suitable gait.

Robot controller To change and find the most suitable gait for a given terrain type, the robot would need to have a robot controller. The robot controller is the brain of the robot, and will give instructions to the different components of the robot. The robot controller will, for example, give instructions to the different joints to be in a certain position. To implement a robot controller there are few requirements to take into consideration. There are five different issues that can affect the design process. These issues are impassable terrain, foot slippage, accidental collision, modeling errors and sensor errors[38]. These are all issues regarding a four-legged robot (which will traverse the different types of terrain) and need to be considered when implementing a robot controller.

One powerful tool used to control robots is a framework called the Robot Operating System(ROS)[47]. ROS can be used to control a variety of different types of robots, from wheeled robots to flying drones. ROS could be used for both controlling a robot in the real world and a robot model in

a simulator. ROS is also very easy to use, because of all the libraries that already exist.

2.2.4 Kinematics

Kinematics is a field of study that gives a geometrical description of a mechanical system. The system can consist of a variety of different components. A component can be a joint or leg, and the components that are connected with each other restrict the robot to move in a certain way. The movements of the different connected components are constrained to each other. Kinematics describes the synergy between the connected components and their constrains, without considering the internal or external forces applied on to the system to create movement.

Defining a mechanical system Each component that constitutes the mechanical system is referred to as a rigid body. It is a rigid body if the distance between any two given points remain constant. The generalized coordinates or configuration coordinates are used to define the configuration of a mechanical system in classical mechanics. Defining the generalized coordinates used depends on the system configuration, and what would make the kinematics equations simpler and more efficient. The generalized coordinates uses a reference coordinate in order to define the configuration of the mechanical system. To be able to define the configuration of a mechanical system, there is a minimum number of generalized coordinates, referred to as the degree of freedom(DoF). DoF is the number of independent generalized coordinates. The reference coordinates, which are commonly used, are Cartesian coordinates, usually referred to as the world frame. All the components will have their own frame and the orientation will be defined by the world frame.

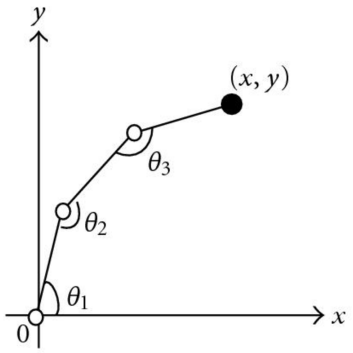


Figure 2.10: Illustration of a simple mechanical system[5].

Inverse kinematics Kinematics is divided into two different approaches: forward and inverse kinematics. In forward kinematics, the joint angles

[5]https://i.stack.imgur.com/TZ58w.png

are specified and used to find a configuration where the position of the end-effector is at the target position. In inverse kinematics, the target position of the end-effector is specified and used to find the configuration of the different joints in the mechanical system. There are few different approaches for solving inverse kinematic problems. Some of the approaches are: pseudoinverse approach[48], Jacobian transpose approach[49] and neural network approach[50]. Inverse kinematics is a bit more complex than forward kinematics, since the inverse kinematics can end up with a result with a few different solutions to the same problem. The reason is that the different joints can have several unique configurations and have a configuration where the end-effector is at the target position.

2.2.5 Stabilization

This section will give an explanation how a robot can look at its own stability in order to traverse a terrain type.

Stabilizing When a legged robot is moving forward, it needs to have the ability to be balanced to avoid falling, or experience unexpected movement. Gaits are divided into two groups, dynamically and statically stable gaits. These are two different strategies in order to remain balanced when walking forward. The difference between the strategies is that dynamically stable gaits are based on movement to remain balanced, whereas static stable gaits rely on being supported by the legs to remain balanced. The strategy chosen is based on the speed. Faster gaits are dynamically stable gaits and slower gaits are static stable gaits.

Dynamically stable gaits In order for a dynamically stable gait to remain balanced it would need to rely on the motion. What this means is when a legged robot is moving forward the robot will be unstable until it maintain stability again by use a leg to catch or prevent the robot from falling. To explain the concept of a dynamically stable gait, passive dynamic walkers[51, 52, 53] is used as an example. In order to move forward and stay stable it needs the help of an incline and gravity. When looking at this example, dynamically stable gait can be described as controlled falling. When the passive dynamic walker is moving forward down the incline it will stop the falling motion by setting down a leg and then start a new falling motion, until it reaches the end of the incline.

Statically stable gaits In order for a statically stable gait to remain balanced the legs, which create a support polygon, should be placed around the center of gravity[45]. A four-legged robot with statically stable gait will need to have a minimum of two legs in contact with the ground to remain balanced. To make it easier to illustrate the basic concept of a statically stable gait, the four-legged robot would need to have a minimum of three legs in contact with the ground to remain balanced. The area in between the three legs will need to include the center of gravity in order

to remain balanced. This area is referred to as the supporting area or a supporting polygon. If the area in between the legs is not including the center of gravity, it may cause the robot to make unexpected movements or even fall over. Without any internal or external force working on the robot, this would be the only factor making the robot capable of remaining balanced. Figure 2.11 illustrates a stable and unstable static gait.

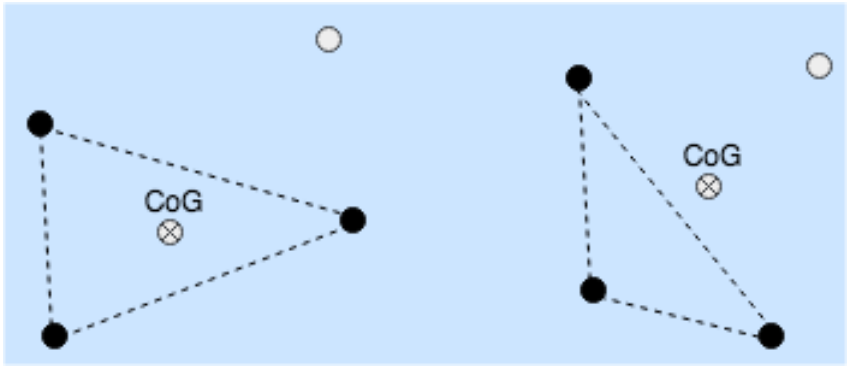


Figure 2.11: Illustration of a stable and unstable static gait. To the left the center of gravity is inside the supporting area, which illustrates a stable static gait. To the right, the center of gravity is outside the supporting area, which illustrates an unstable static gait. The supporting area is marked as three dotted lines, the legs with ground contact are marked as black and the lifted leg is marked as white. The illustration is inspired by McGhee and Frank research on statically stable gaits[45].

Stability on rough terrain Legged robots have the possibility to handle difficult and rough terrain. The issue with legged robots while walking on rough terrain is maintaining stability. When the robot is moving over the terrain, the robot is forced to lift a leg to successfully move forward on the terrain. By lifting one leg, the robot changes the center of gravity and this will effect the stability. In order for the robot to maintain stability, the robot will need to be aware of its own stability to avoid tipping over. There have been different attempts using the stability to design suitable gaits for legged robots on different types of terrains[54].

Orientation and rotation

In order to examine the stability a measurement of the stability need to be introduced. A common method is to examine the rotation of a rigid body. There are different ways of describing the orientation and rotations. Euler angles and quaternions are two commonly used approaches of describing the orientation.

Euler angles Euler angles are three angles that describe the orientation of a rigid body with a fixed coordinate system. The three angles are commonly referred to as:

- Roll
- Pitch
- Yaw

They each represent the rotation around an axes in a coordinate system. Roll is the rotation around x, pitch is the rotation around y and yaw is the rotation around z, shown in figure 2.12.

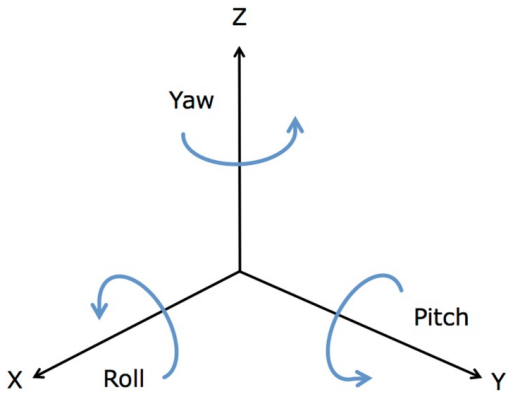


Figure 2.12: Illustration of roll, pitch and yaw[6].

There are two different ways of defining the different rotations, intrinsic and extrinsic rotations. The difference is the use of reference frame, where the reference frame can either be the model or the world frame. Intrinsic rotations are rotations occurring around the axes of the coordinate system attach to the model frame. Extrinsic rotations are rotations occurring around the axes of the fixed coordinate system attach to the world frame.

Quaternions Quaternions is another way of describing the orientation and rotations. Quaternions uses the Euler's rotation theorem, Euler's formula and complex numbers, which gives a convenient mathematical notation that describe a vector $\vec{u}$ and an angle θ.

Euler's rotation theorem explains that any rotation around a fixed point can be represented by an angle θ around an axis, Euler axis, going through the fixed point. The Euler axis is commonly represented by a unit vector $\vec{u}$. Hence, this theorem shows that any rotation in three dimensions can be represented by a unit vector $\vec{u}$ and an angle θ. The quaternions uses four variables to describe the angle θ and the unit vector $\vec{u}$. The fourth variable is set to zero in three dimensions.

[6]https://www.researchgate.net/profile/Suneeta$_G$odbole/publication/262055313/figure/fig2/AS:213872930758658@1428002688513/Average-roll-pitch-and-yaw-angles.png

To represent a unit vector $\vec{u}$ with quaternions, two new complex numbers are introduced, j and k. Quaternions uses the complex number i together with these two complex numbers. Euclidean vector, which is the most common way of representing a unit vector $\vec{u}$, given as (1,2,3) or (x,y,z). However, using quaternions the same vector $\vec{u}$ is given as $1i + 2j + 3k$ or $xi + yj + zi$.

The angle θ is then defined by the unit vector $\vec{u}$ (2.1) and the extension(2.3) of the Euler's formula (2.2)

$$\vec{u} = \left(u_x, u_y, u_z\right) = u_x i + u_y j + u_z k \tag{2.1}$$

$$e^{\theta i} = \cos x + i \sin x \tag{2.2}$$

$$q = e^{\frac{\theta}{2}\left(u_x i + u_y j + u_z k\right)} = \cos\frac{\theta}{2} + \left(u_x i + u_y j + u_z k\right)\sin\frac{\theta}{2} \tag{2.3}$$

2.2.6 Adaptation

This section will present a definition of adaptation and uses of adaptation in robotics.

Definition

Adaptation has been used in different fields of study and therefore has been given different definitions[55]. The definition used in biology is the most general definition. In biology, adaptation is defined as an organism that makes a modification of its own structure or function. The reason why the organism needs to make modification could either be caused by a change in the environment or a change in the morphology of the organism.

In robotics

In the field of robotics there has been completed much research around adaption[56, 57, 58]. The previous research show that a robot could use adaption to adjust to sudden changes in the environment or the structure of the robot. By adapting a robot could, for example, change to a different morphology if the structure is damaged. Hence, a four-legged robot could adapt by using a more suitable gait when there is a change in terrain type. When a robot traverse a certain terrain type the robot could adapt using different gaits. One possibility where the four-legged robot needs to adapt is when change in stability. If the robot is unstable it would need to adapt to a gait where the robot maintain more stability. If the robot is stable it would be possible for the robot to adapt to a gait where the robot maintain more speed. Therefore, with a set of gaits a robot would be able to adapt to different terrain types.

2.2.7 Self-awareness

This section will present a definition of self-awareness, previous work of self-awareness in robotics and lastly explain what makes a robot self-aware.

Definition

In psychology, self-awareness is defined as a psychological state where humans are aware of their own characteristics, feelings and behaviors[59]. This means being able to recognize and separate oneself from the environment and other individuals. In psychology, they also divide self-awareness into private and public self-awareness. Private self-awareness is related to each individual and that an individual is aware of its own private or personal aspects. Public self-awareness is when an individual is aware of their own public aspects, which can be seen and evaluated by other individuals. This means that a self-aware individual is capable of both being aware of itself and how the individual appears to other individuals.

In robotics

In recent years, the concept of self-awareness has been transferred to robotics[60, 61, 62]. By looking at the definition of self-awareness it has much in common with the internal model, presented in section 2.1.1. The private self-awareness can be seen as the robot model and public-awareness can been seen as the world model. Looking at a robot with the internal model, the internal model is a representation of the robot itself and how the robot sees the environment. This makes the robot able to simulate itself and the environment. With this ability, the robot has the opportunity to look ahead to future consequences without completing a given action[63]. A system with an internal model would be capable of "asking" itself what could happen if it perform a certain action. The robot itself could then generate and test a what-if hypothesis[64]. This means with an internal model a robot could ask if it is possible to complete a certain action and what would happen next if the robot chooses a given action. If the robot chooses to complete the action it would generate several possible future actions and then have the ability to chooses the future action from the possibilities generated.

A self-aware robot In order for a robot to be self-aware the robot would need to use the concepts, explained above, properly. When a robot is moving around in the world model it gets feedback from sensors. When the robot gets this information, it would need to generate a robot model and world model. From these two models, it would need to generate possible future actions. When generating the future action, it would need to be looking at the consequences from making a certain action. After evaluating different possible actions, it would need to choose a safe action that would not damage the robot. If the robot is capable of following these steps and

successfully choose a safe action, a robot can be considered self-aware. The steps in order to be a self-aware robot are illustrated in figure 2.13.

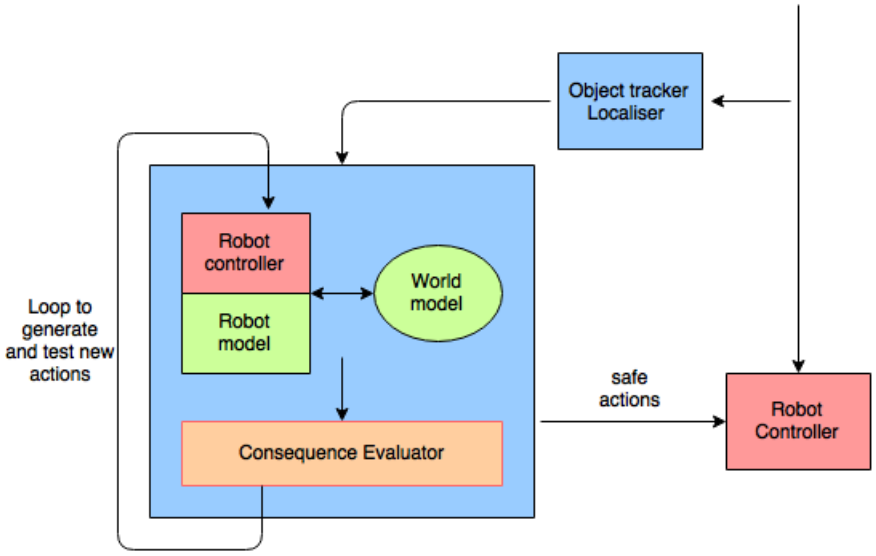


Figure 2.13: Illustrating the steps in a "Self-aware" robot[64]

Chapter 3

Implementation

This chapter explains the different choices that need to be considered in order to make a robot able to predict the most suitable gait. Section 3.1 explains the choice of the implementation environment. Section 3.2 explains the implementation and use of terrain and gaits. Section 3.3 explains the different approaches implemented in this thesis. Finally section 3.4 give an understanding of the data accumulated.

3.1 Implementation environment

The implementation environment consists, as illustrated in figure 3.1, of five parts: language, simulator, a robot, gaits, terrain, and a controller. The next five paragraphs will explain the choices made for each part of the implementation environment.

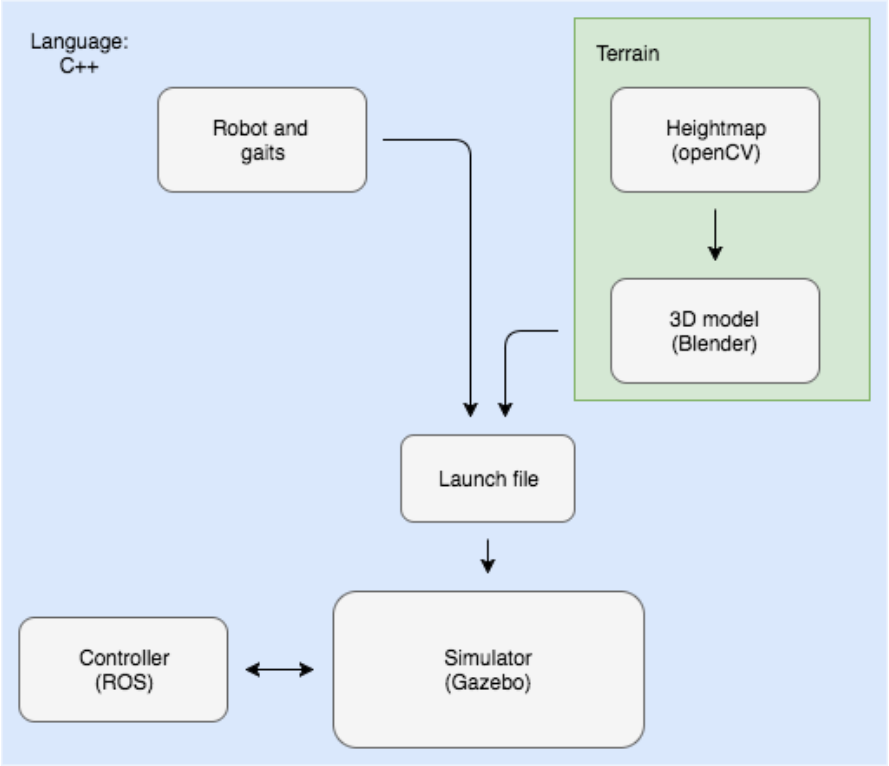


Figure 3.1: Illustration of the implementation environment

3.1.1 Language

In the process of choosing programming language, it was necessary to look at different languages that are available and which libraries they support. By using libraries, the implementation time could be reduced significantly, since the library consists of functions that could easily be reused for the implementation of this thesis. The two programming languages that were considered for this thesis were C++ and Python. The choice of programming language fell on C++. There are several reasons

why C++ was chosen. The main reason was that the robotic group at the University of Oslo has done most of their implementations for prior research and experiments using C++. Some of these implementations are the foundation for the implementation used in this thesis. The second reason is that C++ supports the use of libraries to create the terrain and to control both the simulator and the robot.

3.1.2 Simulator

The simulator itself is the most important component of the implementation environment. In order to end up with valuable and usable data, the simulator needed to be as close to the real world as possible. The different simulators that have been developed in the last couple of years comes with great functionality and the accuracy of the physics calculations are close to the real world. After considering different simulators, the choice of simulator fell on Gazebo. There were a couple of different reasons that made Gazebo the best fit for this thesis. The first reason was that Gazebo had libraries which could be easily accessed with C++. The second reason is the integration with ROS together with C++ and Gazebo, which has also been used by fellow researchers at the University of Oslo.

3.1.3 Robot

The robot used in the simulator is a four-legged robot modeled in Gazebo at the University of Oslo. The reason why an existing robot model was used is that it saved time compared to creating a whole new robot model. The reason for choosing this specific robot is that it is able to have many different gaits, since each leg of the robot has three different joints. In order to get good results, the robot needed to be able to try out different gaits on different terrain types. A good result means that the robot could handle different types of terrains, from flat to rough terrain. All the different joints were modeled, and the different contact points were set, so Gazebo would have the same specifications as the robot in the real world. Figure 3.2 is a screenshot of the robot modeled in Gazebo.

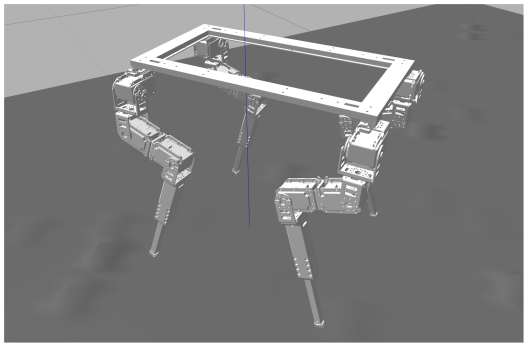


Figure 3.2: The robot used for the simulations. The robot is called Dyret

3.1.4 Controller

ROS(Robot operative system) is an open source library used to control robots[47]. ROS was used to send and receive data from the simulator. This made it easy to control the robot in the simulator and collect data after every simulation.

3.1.5 Terrain

The terrain, like the robot, need to be modeled in the simulator. The different planes and contact points need to be set, so the robot will not be able to fall through the terrain model. The terrain should be modeled in a way that it can be recreated in the real world. To make the terrain for the simulator, a combination of openCV(Open Source Computer Vision)[65] and C++ have been used. OpenCV is a library of programming functions for real time computer vision. Using the library made it possible to manipulate the pixel values in an image. The terrain is generated from heightmaps, which is a gray scale image, where pixel value 255(white) is the highest point, and 0(black) is the lowest point in the terrain. Heightmaps were one of the few different options considered to make the terrain. The reason why this became the chosen solution for creating terrain was the fact that it was simple to generate and less complicated than other approaches, such as Digital elevation model[66]. After the pixel values are manipulated a Gaussian blur smooths the image. The Gaussian blur formula is shown in equation (3.1).

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.1}$$

The reason for using Gaussian blur is to remove sharp edges and end up with an image with smooth transition between high and low point of the image. The aim was to make the terrain more suitable for a robot to traverse. An illustration of a heightmap is shown in figure 3.3.

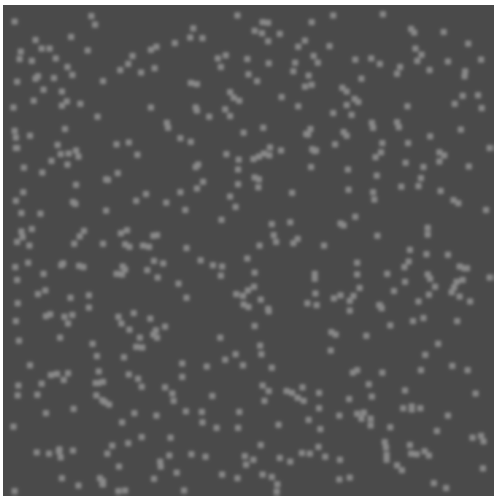


Figure 3.3: Heightmap illustration

The heightmaps used for this thesis were then modeled into 3D using Blender[67]. Blender takes the heightmaps and converts 2D pixel values into 3D coordinates.

3.2 Terrain and gaits

The focus of this thesis is to predict the most suitable gait configuration for a given terrain type. Hence, different types of terrain were created, so that the robot needed to use different gait configurations. This section will presents the experience terrains, test terrain and gaits implemented in the thesis.

3.2.1 Experience terrain

In the real world, for geological reasons, there are many types of terrain. Because of the large amount of terrain types humans have not been able to experience all the different types. Therefore, if they ecounter a new type of terrain, they need to use their past experience to figure out how to handle this new type of terrain. Humans are therefore forced to predict the most suitable gait for a new and unknown type of terrain they are about to walk over. This behavior needs to be transferred into this implementation. This means, when traversing a new terrain type the robot would need to predict the most suitable gait for a certain terrain based on past experiences. Having the robot predict the most suitable gait would then be a good way to show that a robot could be able to predict based on prior experience. To be able to prove this the robot would need to test out different gaits on different types of terrain. Since there is no prior experience used in the traditional approach the experience terrains were only used in the prediction approach.

Different types of terrains There are many types of terrain, and since the robot needs to gain experience, the robot needs to walk over different types of terrain, one of the reasons being that the robot needs to gain experience on how it is to traverse over easy, medium and hard terrain. This would make the robot more capable of predicting the gait when the robot is facing a new terrain type. The experience terrains were designed to be 4x4 meters, to make the robot able to walk a couple of meters without falling off the terrain. After some deliberation, five different terrain types were used. This allowed the robot to experience five terrain types, which also means that the robot could make predictions on these terrain types based on prior experience. By using the robot to make predictions on these five terrain types would be enough to show that the robot has the ability to predict based on prior experience and not being too time-consuming. This was then a trade-off between the number of predictions and time, and became a fair balance between the two factors.

Limitations As mentioned earlier the real world consists of many different types of terrain. In this thesis, the implementation only uses five types of terrain. This makes the robot not able to experience a wide variety of different terrains. Not having a wide variety of experience data will not give the robot the ability to make good predictions on a large amount of different terrain types. This means that the robot could predict a gait which will make it become unstable or even fall. Therefore, having more terrain types that the robot can test out would make the robot capable of making a better prediction on a unseen or a new terrain type.

3.2.2 Test terrain

To make the terrain used for this study, some requirements were taken into consideration. The development of the terrain went through different prototype phases, which resulted in the final test terrain. The first test terrain prototype consisted just of one type of terrain, and the robot was then instructed to predict the suitable gait. The next test terrain prototype was a course consisting of five different terrains. The terrains were assembled in a straight line with flat terrain between each different type of terrain. The flat terrain is an area for the robot to change gait if necessary. After the robot had traversed over this course a couple of times, it was apparent that the course had some flaws and needed to be redesigned. One flaw was that the robot had no option of changing directions and the terrain itself made the robot turn sideways involuntarily because of the structure. The robot would then eventually walk over the edge of the terrain, which counted as a fall, and this had a significant effect on the data collected. The last and final terrain was design as a circular terrain. The robot could now walk sideways without walking off the edge of the terrain. The five different terrain where designed with a circular pattern and attached to each other with a circular flat terrain in between. The different terrain types were designed to be approximately two meters, to force the robot to take a few steps forward. The flat terrain in between the terrain types were intended for the robot to predict the most suitable gait and prepare for that specific terrain type. For the robot to be able to predict the most suitable gait, the robot would need to have different gait configurations. The test terrain consists of the five different types of terrains. The terrain types implemented are a modified version of the five types used as experience terrains. The reason for modifying the different types of experience terrain is to give the robot a better chance of predicting the different terrain types. Exposing the robot to a completely new or unseen terrain will make it difficult for the robot to try predicting the most suitable gait.

Limitations The test terrain consists of five different types of terrain. The order of the terrain types is set to be the same order. To make it more difficult for the robot this order could be randomly selected before each run the robot attempts to traverse the test terrain. Since the robot is only exposed to modified versions of the experience terrains the robot is not exposed to unknown terrain types. The test terrain should be a unknown

test environment, where the robot would need to predict based on the roughness of the approaching terrain type. This will make the robot more independent and not relying on help from human interaction to tell the robot which terrain type it could predict the most suitable gait.

3.2.3 Gaits

As explained in section 3.2.1 the real world, for geological reasons, has many types of terrain. For the robot to be able to traverse the different terrain types it would need to have different sets of gaits. The robot used in this thesis is a four-legged robot and has many unique sets of gaits. For the robot to be able to gain experience, it needs to experiment with different gaits. Therefore, the robot should gain experience by experimenting with traversing different terrain types using different gaits. The speed of a gait should vary, since this is a crucial factor for handling rough terrain. Moving too fast on a rough terrain may cause the robot to trip, get unbalanced or even fall. Hence, the stability will correlate with the change in speed. A slow gait would be more stable, since each step would take more time, a fast gait would be less stable, since it would take a new step before even having completed the previous step. With this in mind, it is necessary to use different gaits, which range from slow and stable to fast and unstable. After experimenting with different types of gaits, the robot needed five sets of gaits to handle the different types of terrains made for the thesis.

Limitations With a four legged robot, there are a numerous number of possible gait configurations. In this thesis, the robot would only use five different gaits in order to traverse a given terrain type. In the real world scenario, it would be optimal to have more gaits, which the robot could try out. If the robot had more gaits to try out it would have the ability to gain more experience and have more options when the robot is trying to adapt to or predict different terrain types.

3.3 Approaches

The two approaches implemented in this thesis are a traditional and a prediction approach. The approaches have both been used to solve the same problem, which is getting the robot from one point to another the fastest way, using a set of gaits. This section will explain how each of the approaches are implemented to solve this problem.

3.3.1 Traditional approach

In this thesis, the traditional approach is used to compare and validate the findings from the prediction approach. Hence, the traditional approach will only get a brief explanation. In this approach, the robot will make a decision of which gait to use based on the stability of the robot when traversing the test terrain.

Execution The robot starts in the middle of the circular test terrain, and moves towards the edge of the terrain. First the robot starts walking forward using the most stable and slowest gait. It then walks until it reaches the flat terrain. While the robot is walking, the robot examines its own stability. If the gait is stable, the robot changes to a faster gait. If the robot is unstable, it uses a slower gait or keeps using the same gait when the gait is the slowest. If the robot is stable enough to move forward it keeps using the same gait. When the robot reaches the flat area, the robot stops, then starts again with the slowest and most stable gait. The robot will then repeat the process for the next terrain type and will continue repeating the process until it reaches the edge of the test terrain. When it reaches the end of the test terrain the robot has completed one out of thirty runs. After every run data is collected and stored. After all the thirty runs have been completed, the collected data from each run is summarized and stored. The process of this approach is illustrated in figure 3.5.

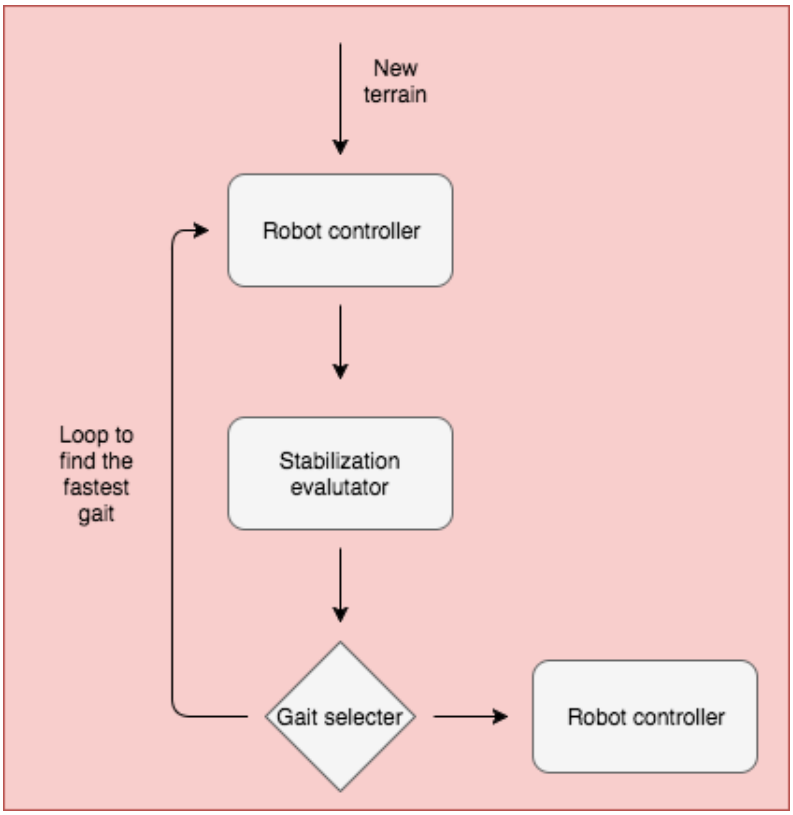


Figure 3.4: Illustrates the traditional approach

Evaluation of stability

To be able to calculate the stability of a robot, there are a few elements to take in to consideration. The method used in this thesis examines the rotation of the robot. The stability is evaluated by the rotation around x, called roll, and y, called pitch, shown in figure 3.5. The rotation around z, called yaw, is not necessary to examine since this rotation describes

the direction of the robot. However, the roll is a representation of tilting forward and backward, and the pitch is a representation of tilting sideways. If the values of roll and pitch are high the robot is unstable and may fall. If the values of roll and pitch are low the robot is stable. To make the robot adapt and use the most suitable and optimal gait the robot would have make an analysis of the values of roll and pitch.

Calculate roll and pitch The calculation of roll and pitch is done using quaternions. The simulator provides information about orientation, which consists of the quaternions. During each run the quaternions are collected from the simulator. When the quaternions are gathered an already implemented function is used to convert the quaternions to Euler angles, which consist of roll and pitch. The roll and pitch are given in radians. After experimenting with the sample rate of the quaternions it ended up to be 2kHz, by taking 100 samples per 0,05 second. The reason is the quick changes of orientation. To handle the quick changes and have a good representation of the gait performance, the samples would need to contain the moments where the robot is unstable as well as where the robot is stable. The average of the 100 samples is calculated from the highest and lowest sample of roll and pitch. The reason is that the robot might be mostly stable during a run, but it might also have certain areas where it struggles. The average of the 100 samples might give the impression that the robot is stable, since the samples with high impact on the roll and pitch will disappear in samples where the impact is low.

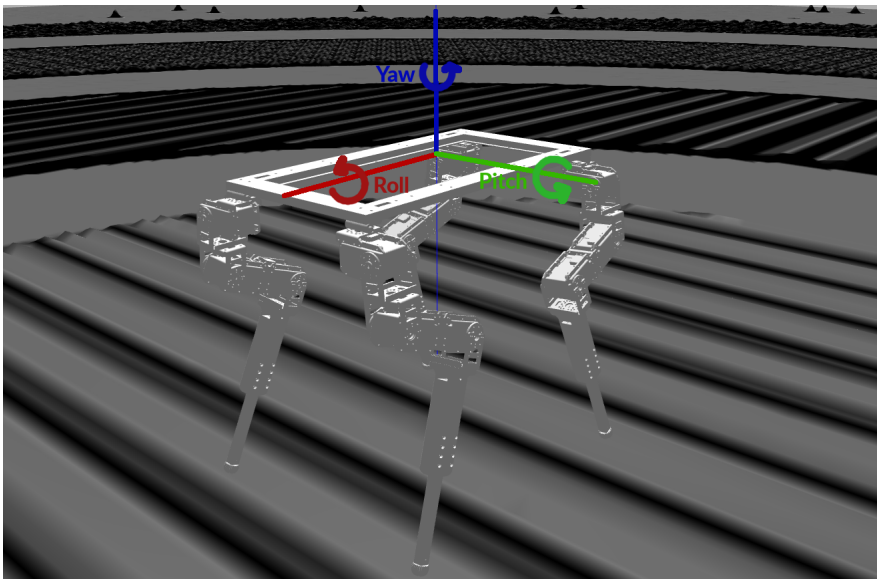


Figure 3.5: Illustration of roll, pitch and yaw on the robot, where the red axis is the x-axis, the green axis is the y-axis, and the blue axis is the z-axis.

Choosing a suitable gait After the data about stability is collected the robot would need to use this data in order to adapt to the terrain and choose the suitable gait. The robot will use a set of thresholds to select the most suitable gait. Threshold is used because there are different cases where the robot would need to make a decision. In the traditional approach, there are three different cases:

- The robot is stable and able to go faster.
- The robot is unstable and needs to go slower.
- The robot is fairly stable and can continue using the same gait.

Finding the right threshold for the different cases was a challenge. First the robot could be fairly stable at one point and then suddenly become unstable. This problem made it difficult to set the threshold where the robot decides to go faster. With a too high threshold the robot was capable of going faster without being stable with the current gait. With a too low threshold the robot would get stuck using the slowest and most stable gait. After experimenting with this threshold, the value ended up to be 0.03 radians for both roll and pitch, when the robot is able to go faster, and 0.06 radians when the robot needs go slower. Algorithm 1 presents the processes of selecting the most suitable gait for the traditional approach.

Algorithm 1 Using threshold to find the most suitable gait

```
1: procedure GAITSELECT(avg_roll, avg_pitch)
2:     if avg_roll > fall_thresh or avg_pitch > fall_thresh then
3:         The robot fell
4:     else if avg_roll < faster_thresh and avg_pitch < faster_thresh then
5:         The robot can go faster
6:     else if avg_roll > slower_thresh and avg_pitch > slower_thresh then
7:         The robot need to go slower
8:     else
9:         The robot keep using the same gait
10:    end if
11: end procedure
```

Limitations In the traditional approach the threshold for when the robot should change gait had a significant effect on the result. The feedback from the simulator indicated that the robot was stable but in a short period of time the robot could be unstable or even fall. To prevent the robot from selecting a gait which was too unstable the thresholds for selecting a faster gait needed be set to a low value. This made the robot select a slower gait most of the time, which had a significant effect on the time used. Another issue was the process of changing gait when the robot could go faster or need to slow down. The transition between the gaits could be more optimal, by eliminating the time preparing for the next gait. In other words, the robot would not need to stop before changing to a another gait.

3.3.2 Prediction approach

This section will present the prediction approach implemented in this thesis.

Execution The robot starts in the middle of the circular test terrain and moves forward towards the edge of the terrain. First the robot will predict the most suitable gait and then start moving towards the flat area. When the robot reaches the flat area, it will stop and predict the most suitable gait for the next terrain, and then walk towards the next flat area. The robot repeats this process until it reaches the end of the test terrain. When it reaches the end of the test terrain, one run is finished, and a new run can start. This continues until it reaches a total of thirty runs. After every run, the data of the robot's performance is collected and stored. After all the thirty runs have been completed the data collected from each run is summarized and stored.

In order to do predictions, the robot needs to use earlier experience to predict the most suitable gait for upcoming terrain. In figure 3.6 is an illustration of how the prediction approach was implemented in this thesis. Section 3.4.1 will give an understanding of the whole diagram, by first give an understanding of the left side of the figure, and then give an understanding of the right side of the figure.

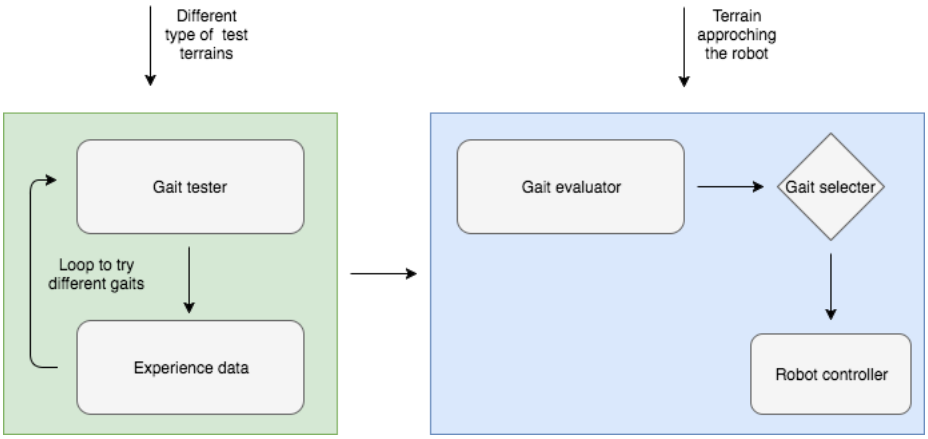


Figure 3.6: Illustrates the prediction approach

Limitations In this thesis, the robot would need to be given the terrain type in order for it to predict the most suitable gait. This means that the robot will know the terrain type which it is approaching. In the real world scenario, this would not be the most optimal solution. In the real world scenario, the robot would need to be able to classify the terrain type using sensors or depth cameras.Telling the robot which terrain type it is approaching will not make the robot able to move around independently. However, this thesis is about predicting the most suitable gait for a specific

terrain type and not classify different types of terrain, which could be a topic for a whole other thesis.

3.4 Data

This section will explain the different data collected, in order to gain experience and to evaluate the performance of the two approaches: traditional and prediction.

3.4.1 Experience data

In order to make a prediction the robot would need experience data. This section gives an explanation of the process of accumulating the experience data and the data accumulated.

Accumulate the experience data

This section will take a deeper look on how the experience data is accumulated. It starts with a new type of terrain and then the robot will walk over the terrain using one gait, then continue until it has tried all the different gaits. After every run, the experience data will be collected. Illustration of the process is shown below in figure 3.7.

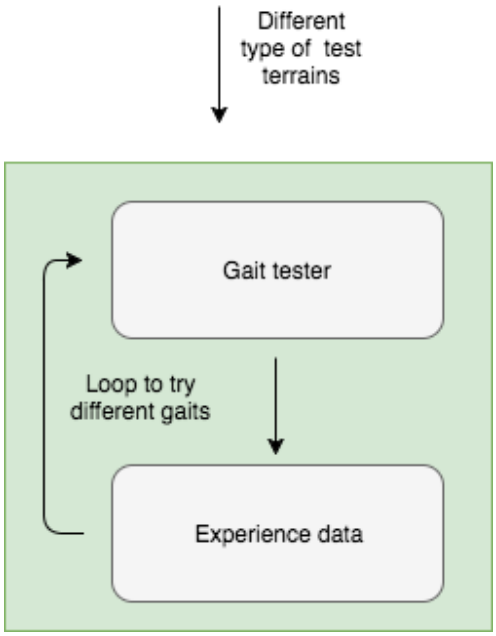


Figure 3.7: Illustrates how the experience data is accumulated

Overall process To accumulate the experience data the robot would need to use all the different gaits on all the different terrain types. This is to gain experience about each gait on the different types of terrain. The robot will then use one gait to traverse the terrain n number of times. It will then

continue using the next, until it the robot has tried all the gaits. When one gait has traversed the terrain type n number of times the experience data is collected. The process of accumulating all the experience data is illustrated in algorithm 3

Algorithm 2 Accumulating experience data for each terrain type

```
1:  procedure COLLECTDATA()
2:      for Number of gaits(i) do
3:          for Number of runs(n) do
4:              Get start position
5:              Start Time
6:              Start gait
7:              Gait stopped
8:              Collect data from current run(n)
9:          end for
10:         Calculate the average of the collected data
11:         Store the calculated experience data
12:     end for
13: end procedure
```

Monte Carlo simulation

In this thesis, the overall goal is to make the robot able to predict and select the most suitable gait with the help of past experience. To gain the needed experience, the Monte Carlo method has been used. The next paragraph will give a better understanding of why the Monte Carlo method has been used in the implementation of this thesis, and what data was expected from it.

Why use Monte Carlo? In this thesis the Monte Carlo approach is used for the learning process. A learning process in this thesis means that the robot will gain experience about each gait traversing different types of terrain. This is one of the reasons for using the Monte Carlo method. The Monte Carlo method collects data based on repeated randomly selected samples. Hence, this means that by using the Monte Carlo method, the robot will walk over the same terrain type n number of times and after every run, the data concerning the gait performance will be collected. Since it collects data from repeated runs, this will give a good representation of how well a robot performed with a certain gait over a given terrain type. A good representation is where the robot is using a certain gait, and covered a significant area of the terrain. This meaning that the robot is not walking on the same area during all the runs. Since the robot is forced to complete n number of runs, using this method, the data will be collected from all the attempts the robot is making. Since the robot is gaining experience from each run, a higher n will give a better result, this is explained further in this

section. This is how we humans, for the most part, gain experience, making the Monte Carlo simulation a suitable choice for this research.

Implementation To be able to implement the Monte Carlo method, there are a few fundamental variables to be taken into consideration. These variables are essential to making the Monte Carlo method work and end up with good results. The variables are listed below.

- Start position
- Walk distance
- Number of runs(n)
- Data collection
- Data storage

Starting point The starting point is set to the edge of the terrain, and the position will vary vertically up or down between every run. The reason it needs to vary is that the robot will then walk over different aspects of the terrain. This will give more general results and not just for a certain area of the terrain. The robot might be able to walk perfectly fine on some areas of the terrain, but might fall on a different part of the terrain.

Walk distance After testing different distances the conclusion was that the robot had to at least walk forward 1-2 meters. If the robot walks less it will not be able to take enough steps, and the result will not be significant. Walking 1-2 meters forces the robot to walk a few steps on the terrain and this will give a good enough representation of the gait performance when it walks over a certain terrain. The reason why the robot is not walking further is to reduce the time per simulation.

Number of runs Then the robot should repeat this step n number of times. After experimenting with different values of n, the robot was set to walk over a certain terrain 200 times with a certain gait. The experimenting consisted of trying different values of n, using a value for n between 100-1000 was shown to give the best result. This will make the robot have 100-1000 different starting positions, which gives good coverage of the terrain. The reason why n was set to 200 is to reduce the time which the robot is using one a certain gait on a given terrain. The higher n is, the more accurate the result will be, but the n should not be too high. Having a high n will just be time consuming and will not affect the result drastically. When choosing the value of n it came down to the trade-off between accuracy and time, getting a accurate representation of the gait performance at the same time not spending to much time collecting the data.

Data collection When the robot is done with a run, the experience data should be collected. This means it needs to collect the data before it restarts a new run. In the implementation, the data from every run is gathered to calculate the mean of how the gait ended up doing. There are many different variables which should be included to get the best representation. In this research, the focus is to look at the basic criteria of a gait and forward movement. The variables included in this basic criterion are:

- Fall percentage
- Distance
- Distance to goal
- Direction
- Time
- Speed

The fall percentage is how many times the robot fell out of all the runs during a simulation. The fall percentage is calculated by dividing the number of falls by the number of runs. In equation 3.2 the calculation, where n is number of runs and x is the number of falls.

$$fall\_precentage = \frac{x}{n} \tag{3.2}$$

Distance is the average distance which the robot has walked during the simulation. This distance should be measured in how far forward the robot has moved, and not consider the direction. The distance is calculated by only looking at the change in x direction. The reason for only looking at the x direction is that the robot is supposed to only walk straight forward. That means that the end point is subtracted from the starting point. The y direction only affect the sideways movement, which will be accounted for when calculating the sideways movement(3.5). In equation (3.3) is the calculation, where x_1 is the start point, x_2 is the end point, and n is number of runs.

$$distance = \frac{\Delta x}{n} = \frac{x_2 - x_1}{n} \tag{3.3}$$

Distance to goal is the average distance from the robot's end position to the goal position. This variable is important since it shows how accurate the gait is. The distance to goal is calculated by subtracting the goal distance, which is a static variable, from the distance walked. In equation (3.4) is the calculation, where D_g is the distance to the goal position, D_w is the distance walked, and n is number of runs.

$$distance\_goal = \frac{D_g - D_w}{n} \tag{3.4}$$

Direction is the average angle which the robot has walked from the starting point. This will show how straight the robot has moved over the

terrain. The angle is calculated from the change in forward and sideways movement, and then takes the arctangent of these changes. The changes are found by subtracting the ending point from the starting point, in both forward and sideways movement. In equation 3.5) is the calculation, where S_m is the change in sideways movement, F_m is the change in forward movement, and n is number of runs. S_m is the change in x direction and F_m is the change in y direction, where x_1 and y_1 are the starting points, and x_2 and y_2 are the ending points. The change in x direction could either be negative or positive depending on right or left movement. Therefore, it was necessary to use absolute value since this is the value of the average angle and having one run with a negative angle of minus five degrees and one run with a positive angle of five degrees would cancel each other and the average angle would be zero.

$$direction = \frac{\arctan\left(\frac{|\Delta S_m|}{\Delta F_m}\right)}{n} = \frac{\arctan\left(\frac{|x_2 - x_1|}{y_2 - y_1}\right)}{n} \tag{3.5}$$

Time is the average time which the robot used to walk over the terrain. The time is calculated by looking at the change in time. That means that the start time is subtracted from the end time. In equation (3.6) is the calculation, where t_1 is the start time, t_2 is the end time, and n is number of runs.

$$Time = \frac{\Delta t}{n} = \frac{t_2 - t_1}{n} \tag{3.6}$$

Speed is the average speed which the robot walked forward on the terrain. Speed is calculated by dividing the distance walked by the time spent. In equation (3.7), D_w is the distance walked (3.3), t is the time spent (3.6) and n is number of runs.

$$speed = \frac{\frac{D_w}{t}}{n} \tag{3.7}$$

Together, these five variables give a good indication of how well a robot walked over a terrain with a certain gait. There might be even more variables that can be included, but to do a gait analysis these variables should be enough to give a sufficient result.

Data storage Finally, the data are stored for further calculations. When the whole Monte Carlo method is done, the data for each run will contribute to average of all the runs. This will be a summary of how well the robot moved with a certain gait over a given terrain type.

Utilize the experience data

As described earlier in this section, the data collected about a gait on a specific terrain is fall percentage, distance, distance to goal, direction, time and speed. This section will take a look at how these variables can decide whether or not a gait is suitable for the terrain given.

Using prediction To make a prediction, the robot would need to rely on the experience data collected from the Monte Carlo method. To make the robot predict based on the experience, it would need to use the gait analysis and in this case, make the prediction before it starts walking on the next terrain. Therefore, the robot will then read the data collected, use gait analysis and then start walking after selecting the most suitable gait for the upcoming terrain.

The overall process After the experience data is collected the robot would need to use the data collected to make a prediction when approaching a new terrain. First the robot will need to evaluate each of the gaits based on the experience data. Then the robot would need to do the actual prediction, which is selecting the most suitable gait for the terrain. Finally, it would use the selected gait to traverse the terrain. This process will be repeated for each of the terrain types where the robot predicts the most suitable gait. A illustration of the process is shown in figure 3.8.

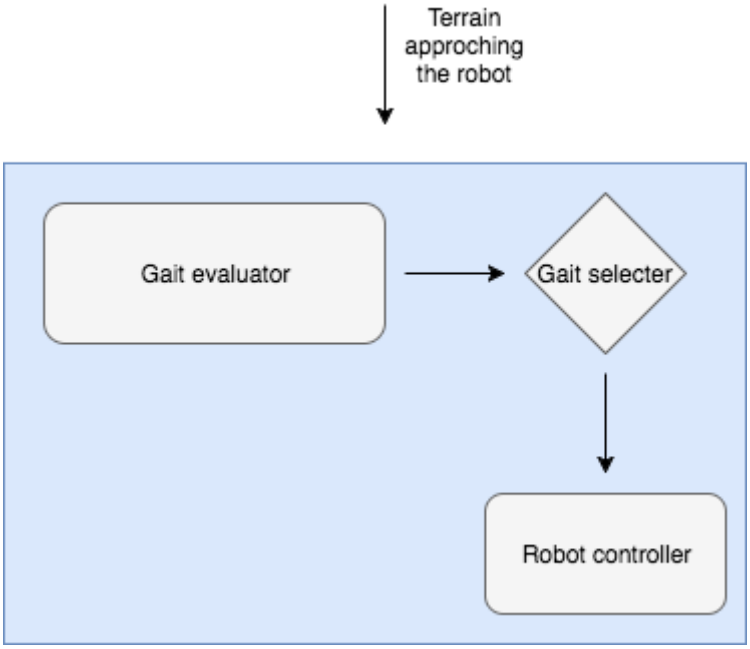


Figure 3.8: Illustrates how the experience data is utilized

Evaluating a gait Analyzing a gait means finding the most suitable gait, but in order to find a the most suitable gait it is necessary to formalize a definition of a suitable gait. For this thesis, the most suitable gait would be the gait that would get the robot from one point to another the fastest way without falling. Depending on the definition, the variables would have different roles. The analysis would then evaluate the gait with focus on achieving this goal. To analyze a single gait, it would be necessary to look at the different data collected. The most significant variable is fall percentage. If this percentage is high, it would not be a good fit for achieving the main

goal. The next variables that have a high impact on the gait performance are the angle and distance walked. If the angle is high, it means that the robot has not walked in a straight line, and this affects the distance walked forward. The distance should be as close to the goal position as possible to be a suitable gait. The last variable to take into consideration is the time. To achieve the goal, the time should be as low as possible. If a gait meets all the criteria above, it will be considered a suitable gait for the terrain.

Comparing the gaits The definition of a suitable gait is now established, but now all the gaits need to be compared. The most important variable the gaits will be compared with is fall percentage. This variable will determine if the gait is going to be used or not. The gait with the lowest fall percentage will be the one the robot would use. However, if the fall percentage is equal, for example 0, we need to look at the other data that is collected from the Monte Carlo method. The second most important is time. Therefore, the next evaluation will need to be time compared with the accuracy. Accuracy consists of two variables, distance and angle, and time consists of speed and time. If the robot were to choose a gait with lower accuracy, the speed should be significantly better. For example, if one gait has an angle of 15 degrees and walked 1.8 meters, where 2 meters is the goal distance, and a speed of 0.5 m/s and time of 3.6 seconds, and the other gait has an angle of 19 degrees, walked 1.7 meters, speed of 1 m/s and time of 1.7 seconds the robot should then use gait number two. Gait number one is a bit more accurate, but gait number two is faster and not extremely inaccurate. This gait would be the most suitable gait to achieve the goal of using the fastest gait without falling. In algorithm 3 is the pseudo code illustrating how he best gait is chosen.

Algorithm 3 Finding the suitable gait in the prediction approach

```
1: procedure FINDBESTGAIT(GAITS)
2:     bestGait ← gaits[0]
3:     checkGait ← ""
4:     for i = 1; i < gaits.size; i++ do
5:         checkGait ← gaits[i]
6:         if checkGait.data.fall_precentage < bestGait.data.fall_precentage
   then
7:             bestGait ← checkGait
8:         else if checkGait.data.fall_precentage == 0
   and (checkGait.data.angle - minDeg) < bestGait.data.angle
   and checkGait.data.distToGoal < bestGait.data.distToGoal
   and checkGait.data.time < bestGait.data.time then
9:             bestGait ← checkGait
10:        end if
11:    end for
   return bestGait
12: end procedure
```

3.4.2 Approach data

This section will explain the necessary data needed to evaluate the performance of both the traditional and prediction approaches. It will first explain the different data needed and how these are accumulated. Then it will explain how these are used to evaluate the performance of the prediction and traditional approaches.

Accumulating the approach

In order to evaluate the performance of the both approaches, a few different aspects from the simulation needs to be taken into consideration. The goal is to make the robot walk as fast as possible from one point to another without falling. Hence, the variables to be taken into account in this study are:

- Fall percentage
- Time
- Speed

The variables listed above are all calculated using the equations presented earlier in this section. Fall percentage is calculated using equation (3.2), time is calculated using equation (3.6), and speed is calculated using equation (3.7).

The choice of variables These variables are all important in order to achieve the main goal. However, the variable that is the most significant is the fall percentage. The reason is that the main focus in this thesis is to make the robot able to traverse the test terrain without falling. Therefore, speed and time will be taken into consideration if the robot has succeeded in completing the whole course.

Limitations The problem with comparing the collected time and speed for prediction and traditional approach is that the prediction approach always has an advantage, because the test terrain is created in order to test the prediction approach. For the traditional approach the robot will always start using the slowest gait for each terrain type. This means that the robot will lose time when it starts traversing a new terrain type compare to the prediction approach. This has a significant affect on the results and when comparing the two approaches.

Chapter 4

Experiments and results

4.1 Experiment setup

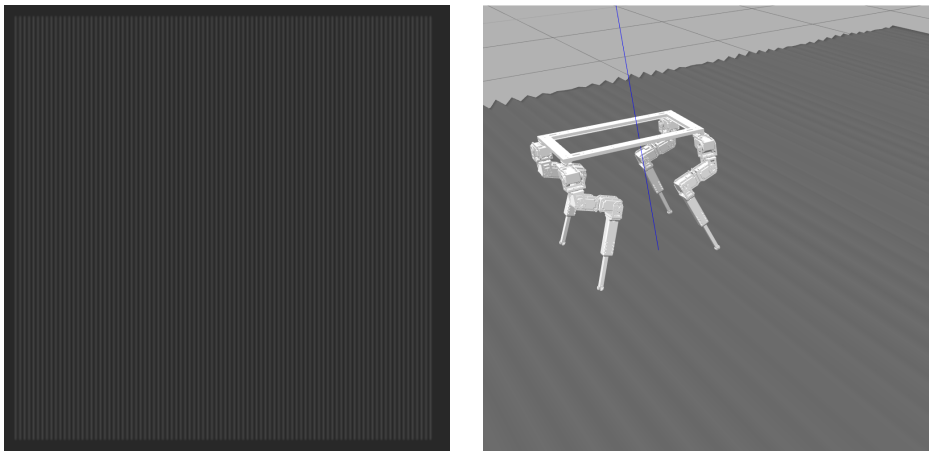
This section presents the different terrain types, the test terrain, and the different gaits used in order to complete the experiments.

4.1.1 Terrain types

To gain experience, five different terrains were generated. The difficulty of the terrains varied from easy to hard, to give the robot a variety of experiences. The five terrains are:

- Horizontal
- Vertical
- Horizontal and vertical
- Random height
- Bump

Horizontal The horizontal terrain type consists of multiple horizontal ridges along the terrain. A screenshot and the related heightmap of this terrain type is shown in figure 4.1. Since the robot need to step over all the ridges it becomes difficult for the robot to move forward. The intention was to make this a semi-hard terrain for the robot.

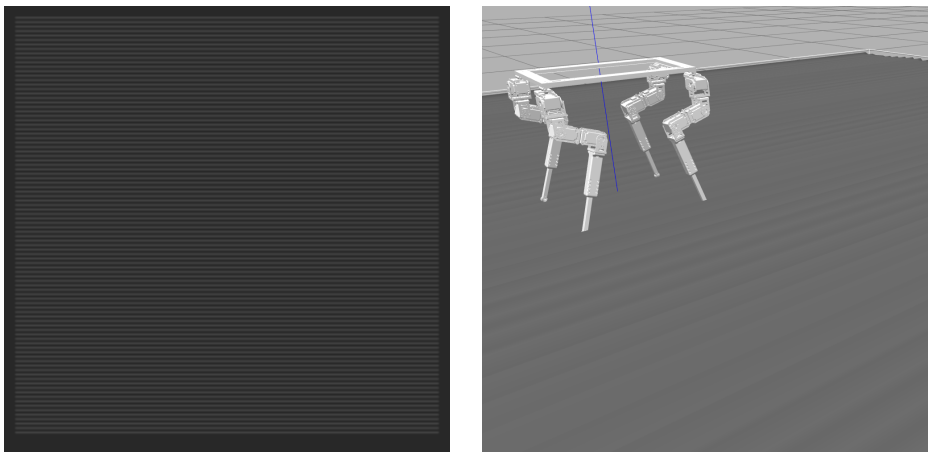


(a) Horizontal heightmap (b) Horizontal terrain

Figure 4.1: In sub figure 4.1a is the heightmap used to generate the horizontal terrain shown in 4.1b

Vertical The vertical terrain type consists of multiple vertical ridges along the terrain. A screenshot and the related heightmap of this terrain type is shown in figure 4.2. When the robot walked on this terrain type it ended up walking in the valleys in between the vertical ridges. This was because when stepping on top of a ridge it would always slide down to the valley

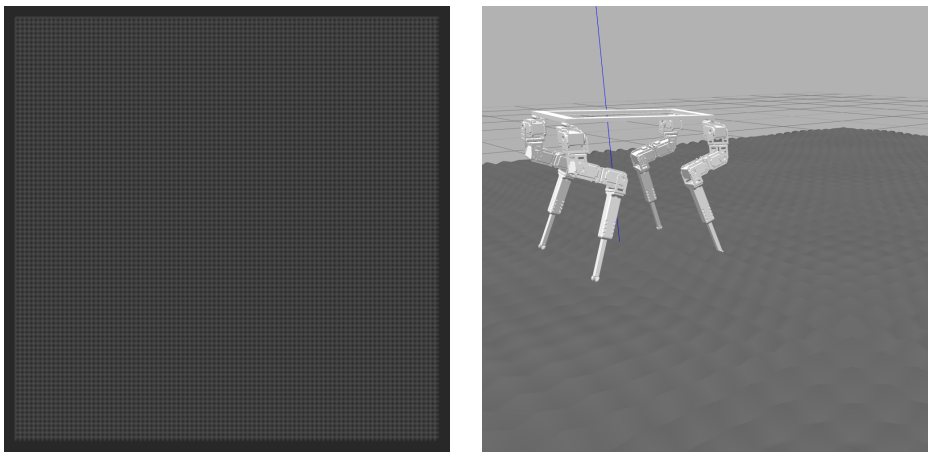
again. This was unintended and made it easier for the robot than expected. Even if this terrain type was easier than expected, it still gave the robot valuable training.



(a) Vertical heightmap (b) Vertical terrain

Figure 4.2: In sub figure 4.2a is the heightmap used to generate the vertical terrain shown in 4.2b

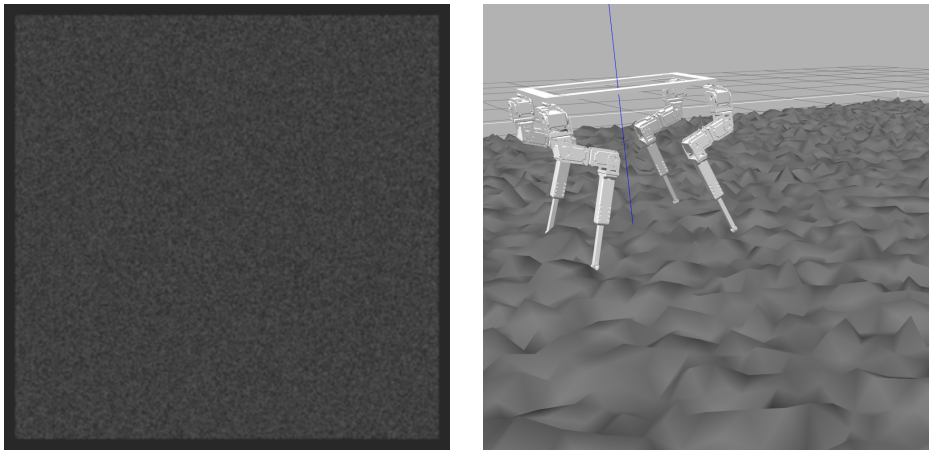
Horizontal and vertical The horizontal and vertical terrain type consists of vertical ridges along with horizontal ridges along the terrain. A screenshot and the related heightmap of this terrain type is shown in figure 4.3. The crossing ridges created small pits that the robot needed to hit in order to be able to move forward. These small pits in the terrain made this terrain type very hard for the robot to move forward. The reason was that the robot was not able to walk on top of the ridges, which made it slide down in between the ridges. When sliding down the robot became unstable at low speed and at high speed the robot could potentially fall.



(a) Horizontal and vertical heightmap (b) Horizontal and vertical terrain

Figure 4.3: In sub figure 4.3a is the heightmap used to generate the horizontal and vertical terrain shown in 4.3b

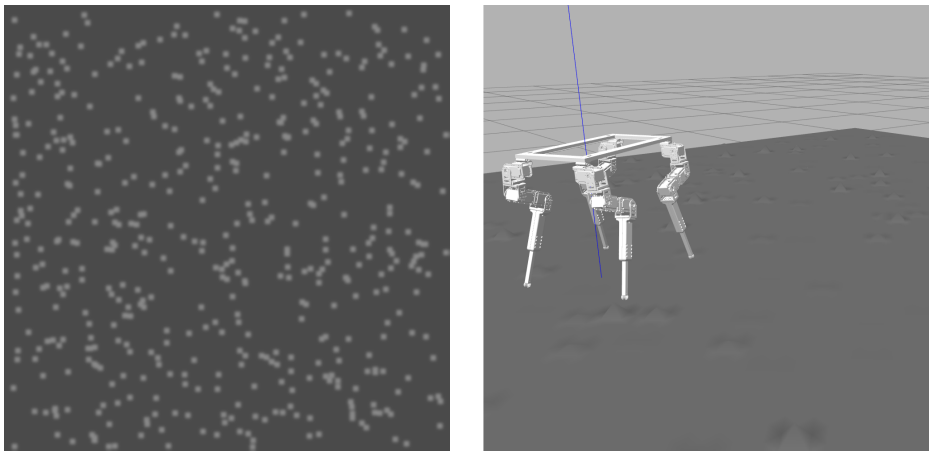
Random Heights The random height terrain type was generated by the height of each pixel of the terrain being random. A screenshot and the related heightmap of this terrain type is shown in figure 4.4. The intention of this terrain was to make it very hard for the robot to traverse the terrain. Since the terrain was all random the robot had great difficulties traversing the terrain.



(a) Random heightmap (b) Random terrain

Figure 4.4: In sub figure 4.4a is the heightmap used to generate the random terrain shown in 4.4b

Bumpy The bumpy terrain type was made by adding random bumps around the terrain. The intention of this terrain was to make it easy for the robot to traverse. Therefore, the number of bumps is set to a low number so the robot does not encounter too many bumps on its way forward. A screenshot and the related heightmap of this terrain type is shown in figure 4.5.



(a) Bumpy heightmap (b) Bumpy terrain

Figure 4.5: In sub figure 4.5a is the heightmap used to generate the bumpy terrain shown in 4.5b

4.1.2 Test Terrain

The test terrain is used in the experiments both for the prediction and the traditional approach. The terrain is a circle terrain as illustrated in figure 4.6. The terrain consists of five different terrain types, which are horizontal, vertical, horizontal and vertical, random heights and bumpy. The five different terrains are a modified version of the five different terrain types described in section 4.1.1. The reason why the terrain types have been modified, is to expose the robot to five slightly different terrain types, and at the same time give the robot a chance to predict the most suitable gait based on the experience data collected previously. In the flat space between the different types of terrain, the robot is able to predict and prepare for the next terrain type. The five different terrain types are designed to be approximately two meters each, and the flat area is design to be one meter each. This will force the robot to walk a few steps on the different terrain types before preparing for the next terrain type, or stop. The flat area does not need to be more than a meter, since the robot is under a meter long, and this area is made to be the area where the robot will change gait. The chosen order of the terrains on the test terrain is first the horizontal terrain, then the vertical terrain, then the horizontal and vertical terrain, then the random heights terrain, and finally the bumpy terrain. The chosen order of the terrain types is random and could have been different from the order chosen in this thesis.

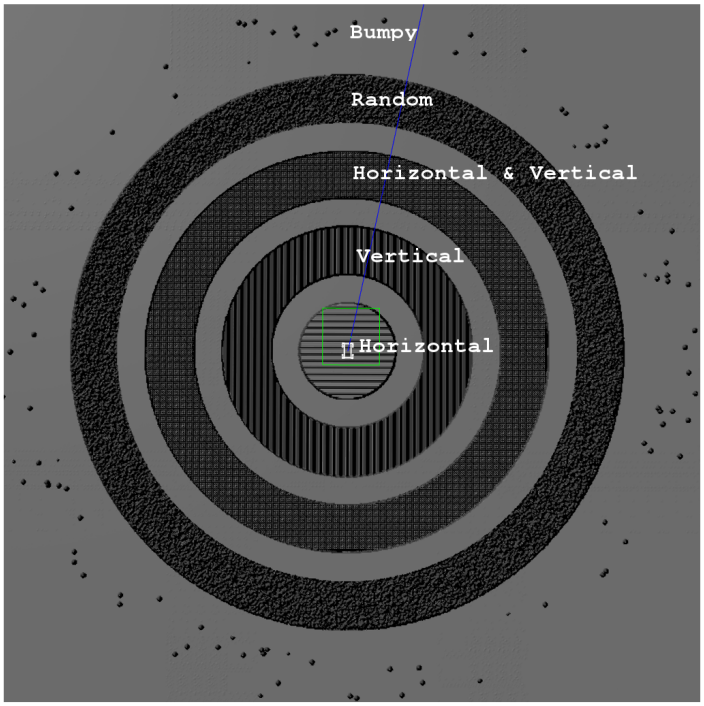


Figure 4.6: Illustration of the test terrain

4.1.3 Gaits

This section will explain the different gaits used in the experiments. It was important to make the gaits in a way that made them more suitable for the terrain types made for the experiments. Having several gaits allows the robot to test different gaits on different types of terrain. The gaits are developed to handle different types of terrain.

Name of the gaits Before presenting the gaits used in the experiments, the name given to the gaits needs to be explained. The gaits used in the experiments are reused from earlier research at the robotics department at University of Oslo[68]. The name given to a gait is divided into four parts. The first part of the name is the method used to generate the gait, which is either multi-objective(MO) or single-objective(SO) evolution. The second part is the objectives used for the optimization of the gait, which is speed and stability for multi-objective evolution, and speed or stability for single-objective evolution. The third part of the name are how many types there is of a certain gait, which is mainly one. The fourth and final part is which individual from the evolution that is selected. This last part is dived into three groups, balanced, stable and fast. Balanced is a gait, where there is a balance between stability and speed. Fast is a gait configuration, where the speed is the individual selected. Stable is a gait configuration, where the stability is the individual selected.

The robot had five different gaits to try on the different terrain types. These gaits varied from stable and slow to fast and less stable. Below is a list of the five gaits, listed from slowest to fastest:

- Gait 1: SO_stability_1_stable
- Gait 2: MO_speedStability_1_stable
- Gait 3: MO_speedStability_1_balanced
- Gait 4: MO_speedStability_1_fast
- Gait 5: SO_speed_1_fast

SO_stability_1_stable This gait is developed to perform at very low speed and very high stability. The robot will be able to use this gait on rough terrain.

MO_speedStability_1_stable This gait is developed to perform at low speed and high stability. The robot will be able to use this gait on rough terrain.

MO_speedStability_1_balanced This gait is developed to perform at an average on both speed and stability. The robot will be able to walk on pretty rough terrain and at the same time move forward with a high speed.

MO_speedStability_1_fast This gait is developed to perform with at a high speed while trying to maintain stability. The robot will be able to use this gait on terrains that are close to flat.

SO_speed_1_fast This gait is developed to perform at a high speed. The robot will be able to use this gait on terrains that are flat and where there are no obstacles in the way. This gait was mainly developed to see how fast the robot could move from one point to another.

4.2 Traditional approach

This section will explain how the traditional approach is tested. It will explain the experiments run to collect the data of the performance of the traditional approach. Then it will present the data collected and analysis the performance from the data collected.

4.2.1 Test execution

In order to test the traditional approach it is necessary to run experiments to collect data and then do an analysis of the performance based on the data collected.

Experiments

How the traditional approach is implemented is descried in section 3.3.1. The experiments regarding the traditional approach use this implementation. Hence, in the experiments the robot will start in the middle of the circular test terrain using SO_stability_1_stable, which is the slowest gait. It would then adapt by using the gaits presented in section 4.1.3. The next gait the robot will use, relative to speed, is MO_speedStability_1_stable, MO_speedStability_1_balanced, MO_speedStability_1_fast, and SO_speed_1_fast. The robot will move forward while adapting until it reaches the flat terrain. Then it will start using the gait SO_stability_1_stable again. The robot will then repeat this process for every terrain type, until it reaches the edge of the test terrain. The whole process from starting in the middle of the test terrain to reaching the edge of the test terrain is considered a run. The robot will then attempt to complete thirty runs on the test terrain. Data will only be collected if the robot is able to complete a successful run. A successful run is when the robot traverses the whole terrain without falling. After a successful run the data will be collected and the simulator will be reset. If the robot is not capable of completing a successful run on the test terrain the simulator will be reset.

Results

The data collected after the traditional approach are fall percentage, time and speed. The table 4.1 gives an overview of the data that was collected.

	Traditional
Fall percentage (%)	40
Time (s)	558.81389
Speed (m/s)	0.017086

Table 4.1: Data collected from the traditional approach

Analysis

To be able to evaluate the performance of the traditional approach the data presented in table 4.1 will need to be examine.

Fall percentage The fall percentage is calculated for the thirty runs where the traditional approach is used on the test terrain. The traditional approach had in total eighteen successful runs, whereas the other twelve runs ended in failure or a fall. This gives a fall percentage of 40 % with the use of the traditional approach.

Time The time spent is a representation of the time used for the robot to traverse the test terrain. The time varied during all the successful runs. The reason is because different gaits were used at different places and that the robot had trouble keeping a straight line, which had a significant effect on the time spent. The varies from 488.848 to 634.422 seconds, which is a significant different. The reason why the time varies is because the robot might use faster or slower gaits from one run to another. Figure 4.7 illustrates the time spent by the robot to traverse each of the eighteen successful runs.

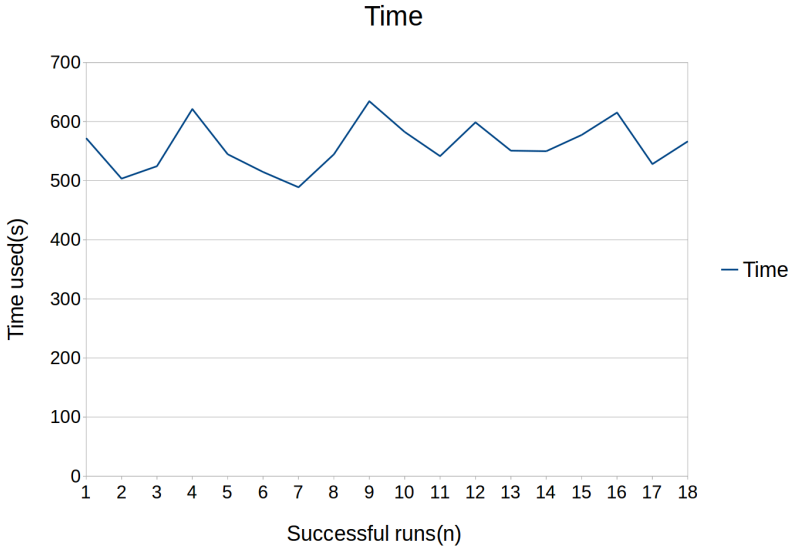


Figure 4.7: Illustration of the time data collected from the traditional approach. The x-axis is the number of successful runs and the y-axis is the speed of the robot over the test terrain.

4.3 Prediction approach

In order to obtain results for the prediction approach the experiments are dived into to gruops: Experience collection and test execution.

4.3.1 Experience collection

This section will present the experiments regarding the learning process of the prediction approach.

Experiments

To collect the experience data the robot starts at the beginning of the terrain and walks towards the end of the terrain. It walks forward two meters and then it stops. When the robot has stopped the experience data from that specific run are collected and stored. The robot repeats the processes until it reaches 200 runs. When the robot is done with all the runs the data collected for each run are summarized and stored.

Results

After running the Monte Carlo method, data were collected and stored. In this section, the experience data will be presented. The tables contain the data collected for all the gaits and how they performed on the different terrain types. The data are collected for the robot walking with a certain gait over the five different terrains. The tables are a summary of all the runs of the Monte Carlo method.

SO_stability_1_stable Table 4.2 presents the data using the SO_stability_1_stable gait. The data shows that this gait performs well on all the different terrain types. The fall percentage is 0 on all the terrains making this the most reliable gait. The gait also has a good accuracy, meaning the robot walks straight and almost ends up at the goal position. The data in table 4.2 reveals that this gait is the slowest gait, and this would have a significant effect on the gait analysis.

Gait	Terrain				
#1	Horiz	Vert	Horiz/Vert	Rand	Bumpy
fall_percentage (%)	0	0	0	0	0
avg_angle (°)	4.353	1.539	2.494	16.747	9.436
avg_dist (m)	0.963	1.428	0.968	0.924	1.595
avg_goal (m)	0.531	0.072	0.532	0.576	-0.095
avg_time (s)	63.669	65.387	63.946	62.316	64.722
avg_speed (m/s)	0.015	0.022	0.015	0.015	0.025

Table 4.2: Data collected using gait #1

MO_speedStability_1_stable Table 4.3 presents the data using the MO_speedStability_1_stable gait. The data shows that this gait performs well on all the different terrains, except on the random heights terrain. The down side with this gait is that the time is high. The time varies from 54.6928 - 57.2954 s, which will have a significant impact on the gait analysis.

Gait	Terrain				
#2	Horiz	Vert	Horiz/Vert	Rand	Bumpy
fall_percentage (%)	0	0	0	1	0
avg_angle (°)	13.754	1.216	4.981	20.747	6.92224
avg_dist (m)	0.69648	1.47282	0.892802	0.838	1.937
avg_goal (m)	0.803	0.0272	0.607	0.662	-0.437
avg_time (s)	55.798	57.295	56.323	54.693	57.216
avg_speed (m/s)	0.012	0.026	0.016	0.0153	0.034

Table 4.3: Data collected using gait #2

MO_speedStability_1_balanced Table 4.4 presents the data using the MO_speedStability_1_balanced gait. The data shows that this gait generally did well on all the different terrain types. The gait struggled a bit more with the horizontal and vertical, and random heights terrain, which is intended to be hard for the robot. The most significant difference in the performance is the avg_angle. This variable varies from 3.06753 - 21.0416 degrees, which has a significant effect on the gait analysis for the prediction approach.

Gait	Terrain				
#3	Horiz	Vert	Horiz/Vert	Rand	Bumpy
fall_percentage (%)	0	0	1	2	0
avg_angle (°)	13.269	3.068	6.630	21.817	9.807
avg_dist (m)	0.847	1.072	0.780	0.833	2.156
avg_goal (m)	0.653	0.428	0.720	0.667	-0.656
avg_time (s)	27.317	27.653	26.983	26.082	28.830
avg_speed (m/s)	0.0310	0.0388	0.029	0.0319	0.0748

Table 4.4: Data collected using gait #3

MO_speedStability_1_fast Table 4.5 presents the data using the MO_speedStability_1_fast gait. The data shows that this gait did poorly on all the different terrains except the bumpy terrain, which was intended to be simple for the robot. The most significant variable is the fall percentage. This varies from 0 - 25 %, which has a significant effect on the gait analysis. The data shows also that this gait uses only 11.0521 - 13.4945 seconds to cover the distance and at same time being fairly accurate, which may have an effect on the gait analysis.

Gait	Terrain				
#4	Horiz	Vert	Horiz/Vert	Rand	Bumpy
fall_percentage (%)	7	9	20	25	0
avg_angle (°)	26.446	14.910	21.725	21.817	13.017
avg_dist (m)	1.089	1.173	1.144	1.164	1.989
avg_goal (m)	0.411	0.327	0.356	0.335	-0.489
avg_time (s)	12.234	12.581	12.227	11.052	13.494
avg_speed (m/s)	0.089	0.093	0.093	0.105	0.147

Table 4.5: Data collected using gait #4

SO_speed_1_fast Table 4.6 presents the data using the SO_speed_1_fast. The data shows that this gait did poorly on all the different terrains. The fall percentage varies from 4-55 %, which will have a significant effect on the gait analysis. The data also shows that this gait is moving very fast over the different terrain types.

Gait	Terrain				
#5	Horiz	Vert	Horiz/Vert	Rand	Bumpy
fall_percentage (%)	44	29	37	55	4
avg_angle (°)	21.686	12.879	17.974	23.021	16.032
avg_dist (m)	1.142	1.226	1.135	1.145	1.956
avg_goal (m)	0.358	0.274	0.364	0.355	-0.456
avg_time (s)	12.292	12.797	12.659	11.021	13.816
avg_speed (m/s)	0.093	0.096	0.090	0.104	0.141

Table 4.6: Data collected using gait #5

Analysis

The robot has now gained the necessary experience to predict the most suitable gait. The robot needs to find the gait which is the fastest, but at the same time is stable enough to get the robot from start to finish.

Evaluate the gait The most significant variables, in order to get the robot from start to finish fastest, are time and fall percentage or chance of falling (COF). Figure 4.8 is an illustration of the time versus the chance of falling based on the data collected in the learning process. It shows how the different gaits perform on the different terrain types, when looking at time and chance of falling.

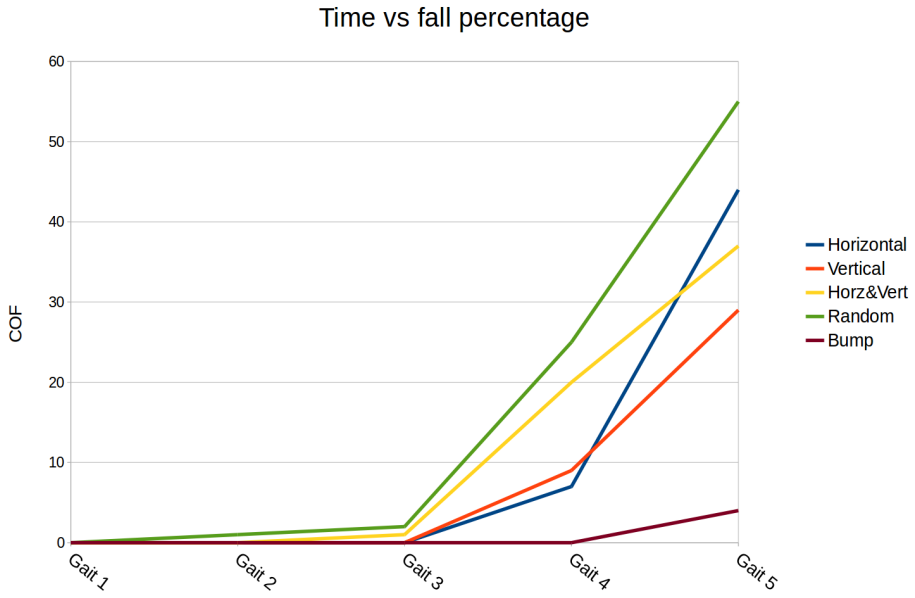


Figure 4.8: The x-axis is the different gaits used in the learning process arranged in an order from the slowest gait to the fastest gait, shown in the list presented earlier in this section

As expected, the fastest gaits performs the worst on all the different terrain types. When the robot predicts the most suitable gait, the graph shows that the robot will not use the fastest gait for any of the terrains. The gaits that are the most suitable to use are the three slowest gaits, SO_stability_1_stable, MO_speedStability_1_stable and MO_speedStability_1_balanced. They all perform well on the different terrains. By following the rules presented in section **??** the robot ended up using the four slowest gaits to complete the circular test terrain. It used the MO_speedStability_1_balanced for both the horizontal and vertical terrain, MO_speedStability_1_stable for the horizontal and vertical terrain, SO_stability_1_stable for the random heights terrain, and finally MO_speedStability_1_fast for the bumpy terrain.

4.3.2 Test execution

In order to test the prediction approach it is necessary to run experiments to collect data and then do an analysis of the performance based on the data collected. This section will first explain the experiments used to collected the necessary data, then it will present the data collected and finally, analyze the data collected.

Experiments

How the predicting approach is implemented is descried in section 3.3.2. The robot start predicting the most suitable gait from the data collected in section 4.3.1. The robot will then predict the most suitable gait for

all the terrain types on the test terrain. As described in section 4.1.2 the test terrain consists of modified versions of the terrain types horizontal, vertical, horizontal and vertical, random height and bumpy. For the simplicity of this thesis the robot will know the order of the terrains. If the robot knows the order it will get the necessary information about the terrain type without any data from sensors or depth cameras. The robot will then predict the most suitable gait for all the different terrain types on the test terrain, until the robot reaches the edge of the test terrain. The process from starting in the middle of the terrain to reaching the edge of the test terrain is considered a run. The robot will then attempt to complete thirty runs on the test terrain. The data will only be collected if the robot is able to complete a successful run. A successful run is when the robot traverses the whole terrain without falling. After a successful run the data will be collected and the simulator will be reset. If the robot is not capable of completing a successful run on the test terrain the simulator will be reset.

Results

The data collected after the prediction approach are fall percentage, time and speed, which is presented in section 3.4.2. Table 4.7 gives an overview of the data that were collected.

Table 4.7: Data collected in the prediction approach

	Prediciton
Fall percentage (%)	16.67
Time (s)	316.17936
Speed (m/s)	0.03049924

Analysis

The collected data need to be analyzed to evaluate the performance of the prediction approach.

Evaluating the performance To evaluate the performance of the prediction approach, the data collect need to be examined. The speed is calculated from the time spent traversing the test terrain. Therefore, the two variables that have a significant effect on the performance are the fall percentage and time.

Fall percentage The first data to be examined is the overall fall percentage for the approach. This is most significant, and will determine if the approach used in this thesis did well or not. The robot fell five times and had twenty-five successful runs of the thirty runs. After all thirty runs the fall percentage was calculated, and as shown in table 4.7, the fall percentage is 16.67 %.

Time The second data set to be examined is the time. This is a variable used to determine if the performance of the approach is efficient or not. Figure 4.9 is an illustration of the time spent by the robot on all the twenty-five successful runs accomplished by the robot. The difference in time for each run is small. The time varies from 276.675 to 356.293 seconds. The reason why the difference fairly small, is because the robot uses the same gaits every time it traverses the test terrain. After all twenty-five successful runs, the time is calculated and as shown in table 4.7 the speed is 316.17936 m/s.

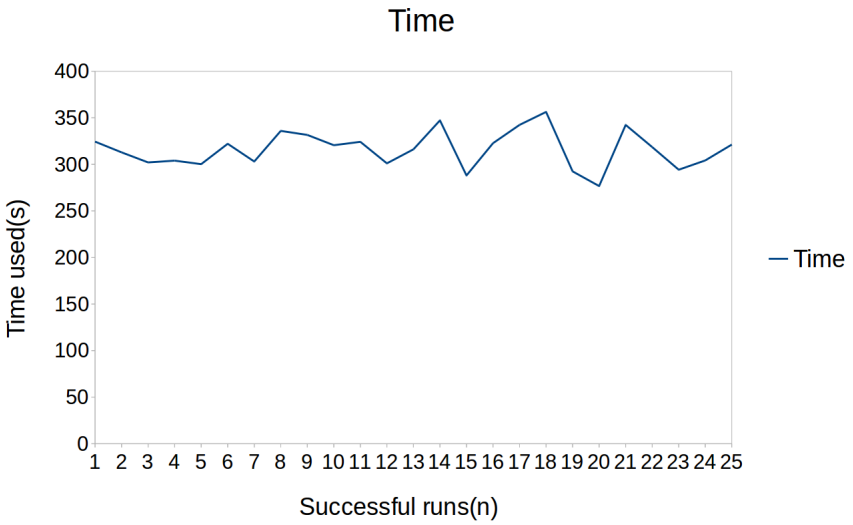


Figure 4.9: Illustration of the time data collected from the prediction approach. The x-axis is the number of successful runs and the y-axis is the time of the robot over the test terrain.

4.4 Evaluating the prediction approach

To evaluate the prediction approach it needs to be compared with the traditional approach implemented in this thesis

4.4.1 Comparing results

Having presented the results from both the traditional and prediction approaches, the performance of the two approaches can be compared. The data collected for both the traditional and prediction approaches are shown side by side in table 4.8.

Table 4.8: The data from both the prediction and traditional method

	Prediciton	Traditional
Fall percentage (%)	16,67	40
Time (s)	316.17936	558.81389
Speed (m/s)	0.03049924	0.017086

4.4.2 Analysis

To compare the performance of the traditional and prediction approaches the fall percentage was the first data to be compared and secondly the time used to traverse the test terrain.

Fall percentage As mentioned earlier, the fall percentage is the most significant variable used to measure the performance of the specific approach. Table 4.8 clearly shows that the prediction approach is a better approach for traversing the test terrain. The difference between the prediction and the traditional approaches is significant: the traditional approach has a 23,33 % greater chance of falling.

Time Time was used as another variable to measure the performance, when the robot was able to successfully complete the test terrain. With the eighteen successful runs the traditional approach had an average time of 558.81389 seconds, whereas the prediction approach had an average time of 316.17936 seconds. The difference in time also show that the prediction approach is significantly better than the traditional approach used in this thesis. In figure 4.10 is an illustration of the time spent on each of the successfully runs in both the traditional and prediction approaches. This figure shows that the prediction approach has a general lower time usage, then the traditional approach.

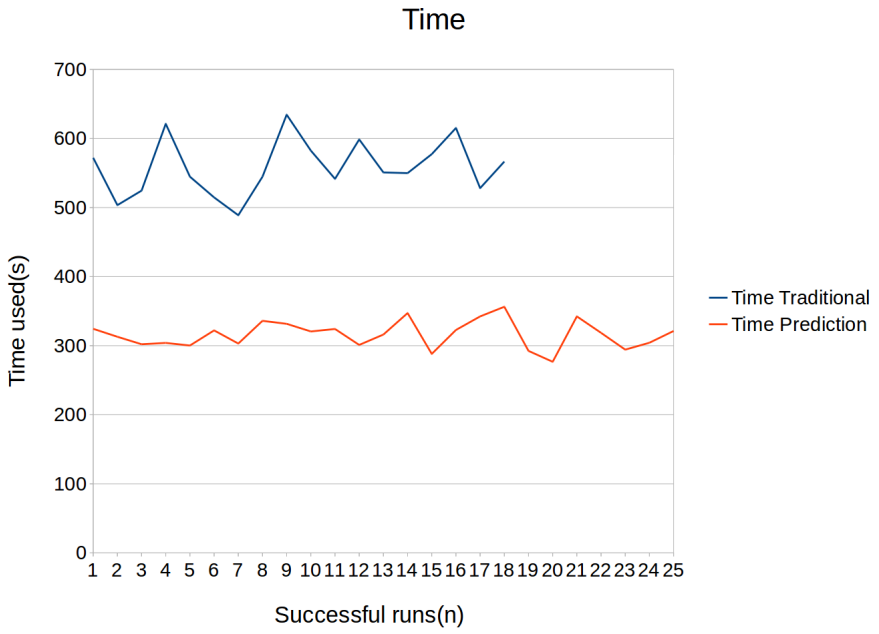


Figure 4.10: Illustration of the time data collect from both the prediction and traditional approaches. The x-axis is the number of runs and the y-axis is the time used by the robot to traverse the test terrain.

Chapter 5

Discussion

5.1 Utilizing the Monte Carlo method

This section will look into and discuss the use of the Monte Carlo as a method for accumulating the experience data.

5.1.1 Data quality

In order to predict the most suitable gait based on past experience there are certain requirements to the data quality. The gained experience data would need to contain each gait's performance and establish a significant difference of each gait.

Accumulated data As presented in section **??**, the data collected were fall percentage, distance, distance to goal, direction, time and speed. The significant of the different variables varies and some of the variables were not even used to select the most suitable gait. After running all the experiments it turned out that the four variables having an effect on the selection of a gait were; fall percentage, distance to goal, direction and time. These were the only variables needed to be able to select the most suitable gait. The variable direction was not needed, since the variable distance to goal gave a better representation of how accurate the robot was compared to the actual distance. However, the distance was needed to calculate the distance to the goal. The speed and time are two variables representing the same measurement. After taking this into account it is not necessary to collect and calculate both variables. It is only necessary to use time to present efficiency of a gait. By using these four variables it gives a representation of the performance and gives the robot the ability to establish a significant difference between each of the gaits used in the thesis.

The ideal data The data was collected by running the Monte Carlo method with the number of runs set to 200. This produces a probability distribution of the chance of falling using a gait on a terrain type. The Monte Carlo method produced a probability distribution for the chance of falling, but this distribution might not be the most ideal. Running the Monte Carlo method with a higher number of runs could have given a more ideal probability distribution. If the number of runs was set to 1000 it might produce a result closer to the ideal chance of falling. This can be illustrated with a simple experiment using a coin which consists of heads and tails. The ideal probability distribution is 1/2 for both heads and tails. If the coined is flipped 100 times it might give probability distribution where there is a 2/3 possibility of getting heads and 1/3 possibility of getting tails. If continuing flipping the coin, the probability distributing will get closer to the ideal 1/2 chance of landing on both heads or tails. The same will also be for the probability distribution for the chance of falling, hence why a higher number of runs could have given a more ideal probability distribution.

number of runs The trade-off between time consumption and the quality of the data became a consideration when implementing the Monte Carlo method. By increasing the number of runs it also increases the time usage drastically. Each run consisted of the robot traversing a terrain type, hence the runs were time-consuming by itself. After considering the selected number of runs that makes up the probability distribution produced, this gave the indication that the quality of the data was precise enough. This meaning that the robot had the ability to use the data accumulated to make valid gait predictions on the test terrain used in this thesis.

5.1.2 Experience data

To give the robot the chance to predict on different types of the terrain it would need experience from using the gaits on these types of terrain.

Number of gaits and terrains The Monte Carlo method is based on using high number of runs to produce a probability distribution. This comes as mentioned with a trade off with the time consumption and how many different terrains and gaits that can be used. Recalling section 3.2 the number of different gaits and terrain types were set to five. By using the Monte Carlo method, the robot would then use all five gaits on one terrain type and then do the same for all the different terrain types. In total this is 5000 runs, where each run is time-consuming by itself. By adding a new gait or terrain type it will then add 1000 runs to the total. The ideal scenario would be to have the robot gain experience from as many gaits and terrain types as possible. Accumulating experience data using the Monte Carlo method is time-consuming, which then puts a constrain on the number of different gaits and terrain types.

5.1.3 The viability of the Monte Carlo method

The overall section has so far discussed the results obtained by using the Monte Carlo method. This section will discuss if this makes the Monte Carlo method a viable method for gaining experience.

For the intended use in this thesis the Monte Carlo method has shown to be a viable method of gaining experience. The Monte Carlo method forces the robot to traverse different terrain types using different gaits, n number of time. This gave a good representation of the performance of each gait on the different terrain types. However, using the Monte Carlo method was a very time-consuming method to use. For each of the runs(n) the robot needs to walk over the terrain in the simulator. Running a Monte Carlo method could therefore end up taking a few hours in order to complete all the runs(n).

5.2 Evaluation of the prediction approach

This section discusses the performance of prediction approach and the comparison with the traditional approach implemented in this thesis.

5.2.1 Performance

The discussion on the performance of the prediction approach is based on the result obtained in section 4.3.

Data quality As presented in section 3.4.2, the data accumulated in order to evaluate the prediction approach is fall percentage, time, and speed. The most significant variable is the fall percentage. Time and speed are two variables for the same measurement. It would only have been necessary to use one of them. This means using the fall percentage and time made it possible to distinguish whether or not the prediction approach performed well. However, to give a more complex or detailed evaluation of the performance, the number of variables should have been increased. It should at least be a measurement for direction and distance. The problem with introducing these two variables into this experiment were that the robot itself did not have the ability to change direction or turn. Since the robot does not have this ability the direction and distance factor becomes irrelevant.

The viability of the prediction approach The results in this experiment give an indication that the prediction approach is a viable approach for traversing an environment consisting of different terrain types. Specifically, the fall percentage obtained in the experiments is a proof that prediction could be used as viable approach. However, it would be dependent on the usage of the robot. If the robot, for example, needs to be able to move around on Mars, it would be important that the robot could move around without falling. This is because on Mars it would most likely be on its own, without humans having an easy access to it. If the purpose on the other hand, was for the robot to walk around on patrol in a building it would be less important that the chance of falling is close to zero. This is because it is easier to access the robot and have humans help it back on its feet.

5.2.2 Comparison

This discussion on the comparison done between the two approach implemented in this thesis, is based on the result obtained in section 4.4.

Comparison data As presented in section 4.4, both approach gather the same variables in order to evaluate each approach separately. With the accumulated data, it is possible compare the performance of the prediction approach with the traditional approach. However, the problem with the comparison is the same problem as mentioned for the data accumulated

for each approach. In order to make a more complex and thorough comparison, the data would need be more detailed.

The viability of the comparison The results presented in this section clearly shows that the prediction approach performs better than the traditional approach. The prediction approach has a better fall percentage and uses less time in the process of traversing the test terrain. However, the prediction approach has a better starting point, since the test terrain were designed in order to test the prediction approach. The result on the traditional approach could have been improved by designing a test terrain just for testing this approach. The intention of only designing one test terrain was to give both approaches the same starting point. After testing both approaches on the test terrain, this worked against its purpose, giving the prediction approach a much better starting point.

5.3 Prediction

This section will discuss the usage of prediction and how it can be beneficial in certain situations.

5.3.1 Additional functionality

For the purpose of this thesis the prediction approach was implemented to work independently. This was to prove that prediction can be used to find the most suitable gait. After looking at the implementation and the results obtained, it indicates that prediction should be used as an additional functionality to a more tested and stable approach. The most commonly used approach is where the robot adapts while traversing the terrain, like the traditional approach implemented in this thesis. In order to make a prediction the robot needs to experience the terrain type first. This is a time-consuming process and will be more beneficial to do while the robot is traversing different terrain types using an traditional approach. When there is an unexpected change in terrain type, the robot could use the gained experience to predict the most suitable gait. A robot will always need experience to use prediction and it is therefore more beneficial to use prediction as additional functionality.

5.3.2 Use of prediction

In this thesis, the results show that it can be used to predict the most suitable gait. The results obtained in this thesis indicates that there are possibilities of using prediction in other areas. This is because a robot will always collect data when performing a task or action. By storing this data gives the robot experience data. When an unexpected change occurs, the robot could use the collected data to predict the most suitable outcome. One area being where the robot gain experience by doing a repetitive task. One example is that the robot can predict the most optimal route when

patrolling in a building. The robot gains experience while walking on patrol and when there is an unexpected change the robot can predict the most suitable route given a certain situation.

Chapter 6

Conclusion and future work

6.1 Conclusion

This study aimed to determine whether or not prediction can be used to predict the most suitable gait for a given terrain based on prior experience. This section will attempt to draw a conclusion from the results obtained in the thesis.

The technology used in the field of robotics are relatively new, and because of that the technology is constantly developing. Prediction has existed in animals as long as we are aware of civilization. What both technology and prediction have in common is that there is a great deal to learn about them. This thesis is a step in the direction of showing that prediction can be used in robotics. The results provided in this thesis, is a small part of the possibilities that are within the field of prediction in robotics. However, the results obtained in this thesis shows that using prediction can be a viable asset to the field of robotics.

This research has shown that prediction can be used to predict the most suitable gait based on prior experience. Using the prediction approach the robot only had a 16.67 % chance of falling. This percentage is fairly low considering that a robot always will have a chance of falling. However, all the experiments are executed in the simulator, which might not reflect the real-world scenario. In order to test the prediction approach in real-world scenario it would need to be tested on the physical robot. There are also still improvements to be done with the prediction approach implemented in this thesis, which will be presented in section 6.2.

After comparing the prediction approach with the traditional approach it is clear that the prediction approach performs better when it comes to traversing new terrain type. However, since this thesis aimed to test the use of prediction, the terrains were designed for the prediction approach. This gives the prediction approach an advantage over the traditional approach. With that being said, the learning process of the prediction approach is time-consuming and therefore need a long period of time to gain experience. Whereas the traditional approach can start traversing the terrain without having gained experience. If the learning process of the prediction approach is considered as part of the total time, the traditional approach is by far better to use.

After obtaining and analyzing the results, it indicates that it will be better to use prediction as an additional functionality. What this means is gaining experience while using a traditional approach. Then when an unexpected change in the environment happens the robot could use the experience to predict the most suitable action. It can be used to predict moving object in the environment, for example, predict where a moving ball will end up at a certain time. The robot would then be able to use this prediction to, for example, move away from the ball or catch it. Therefore, using a traditional

approach with prediction will give the robot the ability to learn and gain experience while moving around. This will eliminate the time-consuming processes of gaining experience before traversing a terrain.

Overall the prediction approach in this thesis provide a result showing that prediction can be beneficial in robotics. It can be used as the main approach for traversing different terrain types. However, it would be more convenient to use prediction as additional functionality to a traditional approach. The use of prediction in robotics is a fairly new field in robotics and it will be interesting to see how it can improve robots in the future.

6.2 Future work

This section will suggest possible improvements of the implementation for future work and possibilities for the prediction approach in future work.

6.2.1 Simulator improvements

When running the experiments in the simulator, there were some different complications, which affected the results. The issues had to do with the controller, the terrain and the robot.

Terrain improvements There were two main problems with the terrain. The first problem with the test terrain was that there were a couple of spots on the terrain were the contact points appeared to be wrong, which made the robot unstable or fall. It would be necessary to eliminate the wrong contact points. Eliminating the wrong contact points will make it closer to the real world. The second problem with the test terrain was the edge between the flat and the different terrain types. In order to start the terrain, the robot was forced to take a higher step to begin traversing the new terrain, which made the robot unstable or fall. It would be a better idea to have the flat terrain and the highest point of the different terrain types on the same plane. Having the different types of terrain on the same plane will eliminate the need for the robot to make a higher step when it begins to traverse the different terrain types.

Robot improvements The problem concerning the robot is the ability to change direction. In all the experiments the robot has not the ability to change direction. This had an affect on the design of the test terrain and the data accumulated for both approaches. Since the robot did not have the ability to change direction it could not straighten up after unexpected change of direction. It would be necessary to use a robot that has the ability to change direction. This will be beneficial for both the viability of the data accumulated and the test terrain.

Controller improvements There were two problems regarding the controller. The first problem with the controller was that it suddenly, during

a run, lost contact with the simulator. When the controller lost contact the robot stopped walking forward, which made the run not count as a valid run. The second problem was that after a successful run the robot was not able to reset to the starting position. These problems made it impossible to complete all the runs at once. All the runs are therefore run one by one, which made it a time-consuming process to collect the data for both the prediction and traditional approaches. It would be necessary to make changes to the controller in order to make the robot traverse the test terrain both without losing contact with the controller and reset to the correct starting point. Implementing these improvements would make it possible to do more runs and therefore get a more ideal result.

6.2.2 Traditional approach improvements

The traditional approach, as described in section 3.3.1, is based on the stability of the robot. The problem is getting the right measurement for when the robot is off balance and when it is stable. In the implementation in this thesis the data showed that the robot was stable most of the time. However, observing the robot in the simulator clearly showed that the robot was off balance. The data used in this thesis is an average of a number of samples and by taking the average, the spikes in stability would be smoothed out, and hence not showing that the robot was off balance. A recommendation for future work, is to take fewer samples more frequently than done in this experiment. This is because when taking samples more frequently the spikes are more likely to show up and give a better representation of the stability. It would also be beneficial to create a new test terrain that fits the traditional approach better than the one used in this experiment. By designing a new test terrain, that works better for the traditional approach, it would give a more fair comparison between the two approaches used in this thesis.

6.2.3 Prediction approach improvements

The prediction approach implemented in this thesis have mainly to improvements, gaining experience and terrain classification.

Experience improvements

As described in section 3.2, the robot only gains experience by using five different gaits on five different terrain types. By increasing the number of gaits and terrain types the robot would gain additional experience. With additional experience the robot could have the possibility of making prediction on terrain never experienced, instead of using modified versions of the experienced terrain types. Another improvement is to give the robot the ability to gain experience while traversing the test terrain, instead of having a separated learning phase as done in this experiment. This would especially be a good idea when the robot is traversing new terrain types

that it has never experienced before. This would save time and at the same time improve future predictions.

Terrain classification

In the implementation done in this thesis the robot received information about terrain type ahead of it. It will be necessary to make the robot be able to classify the next terrain type that it is approaching. In order to accomplish this, the robot will need to use sensors and depth cameras. Giving the robot the ability to classify the next terrain type means it will be able to move around without human interaction. One suggestion is to implement a roughness scale to classify the terrains. When a robot collects data about the terrain type ahead it would then use the data to calculate a roughness factor, which then can be compared with past experience. Then, from the experience data it would know which gaits that works best on different types of roughness. The robot might then be able to predict the most suitable gait based on the roughness of the terrain type.

6.2.4 Real world testing

This thesis aimed to make an approach to predict gaits based on prior experience. The approach has only been tested in a simulator and the results might not be representative of a real world scenario. After making some improvements, it would be necessary to use the prediction approach on the physical robot. Having tested the prediction approach in a simulator gives an indication of how the robot can work in real world scenario, but to be more certain it needs to be verified also in the real world.

Bibliography

[1] M. Raibert, K. Blankespoor, G. Nelson, R. Playter, and T. B. Team, "Bigdog, the rough-terrain quadruped robot," in *Proceedings of the 17th world congress*, vol. 17, pp. 10822–10825, Proceedings Seoul, Korea, 2008.

[2] D. M. Wolpert and J. Flanagan, "Motor prediction," *Current Biology*, vol. 11, no. 18, pp. R729 – R732, 2001.

[3] H. Mannila, "Data mining: machine learning, statistics, and databases," in *Proceedings of 8th International Conference on Scientific and Statistical Data Base Management*, pp. 2–9, Jun 1996.

[4] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[5] S. Madden, "From databases to big data," *IEEE Internet Computing*, vol. 16, pp. 4–6, May 2012.

[6] S. Lohr, "The age of big data," *New York Times*, vol. 11, no. 2012, 2012.

[7] M. Usmani, S. H. Adil, K. Raza, and S. S. A. Ali, "Stock market prediction using machine learning techniques," in *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, pp. 322–327, Aug 2016.

[8] R. R. Halde, "Application of machine learning algorithms for betterment in education system," in *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, pp. 1110–1114, Sept 2016.

[9] P. Suryachandra and P. V. S. Reddy, "Comparison of machine learning algorithms for breast cancer," in *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 3, pp. 1–6, Aug 2016.

[10] H. Li, "An approach to improve flexible manufacturing systems with machine learning algorithms," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pp. 54–59, Oct 2016.

[11] G. T. Tchendjou, R. Alhakim, E. Simeu, and F. Lebowsky, "Evaluation of machine learning algorithms for image quality assessment," in

2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 193–194, July 2016.

[12] G. L. Saboya, O. L. G. Quelhas, R. G. G. Caiado, S. L. B. Franca, and M. J. Meirino, "Monte carlo simulation for planning and decisions making in transmission project of electricity," *IEEE Latin America Transactions*, vol. 15, pp. 431–438, March 2017.

[13] W. Chun, L. Kexin, H. Mingliang, and D. Jing, "On the radar detection analysis under the interference environment based on the monte carlo method," in *2016 11th International Symposium on Antennas, Propagation and EM Theory (ISAPE)*, pp. 492–495, Oct 2016.

[14] S. Wang, L. Zhou, Y. Li, and J. Wu, "Parallel monte carlo method with mapreduce for option pricing," in *2016 IEEE Trustcom/BigDataSE/ISPA*, pp. 2116–2121, Aug 2016.

[15] E. Harstead and R. Sharpe, "Forecasting of access network bandwidth demands for aggregated subscribers using monte carlo methods," *IEEE Communications Magazine*, vol. 53, pp. 199–207, March 2015.

[16] C. Z. Mooney, *Monte carlo simulation*, vol. 116. Sage Publications, 1997.

[17] M. S. Norouzzadeh and J. Clune, *Neuromodulation Improves the Evolution of Forward Models*. 2016.

[18] M. Dulebenets, R. Moses, E. Ozguven, and A. Vanli, "Minimizing carbon dioxide emissions due to container handling at marine container terminals via hybrid evolutionary algorithms," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2017.

[19] S. Strasser, J. Sheppard, N. Fortier, and R. Goodman, "Factored evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 21, pp. 281–293, April 2017.

[20] N. Yadav, V. Yadav, and P. Verma, "Role of evolutionary algorithms in software reliability optimization," in *2016 International Conference System Modeling Advancement in Research Trends (SMART)*, pp. 45–48, Nov 2016.

[21] T. F. d. Q. Lafetá, M. L. d. P. Bueno, C. R. S. Brasil, and G. M. B. d. Oliveira, "Many-objective evolutionary algorithms for multicast routing with quality of service problem," in *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 187–192, Oct 2016.

[22] Bharti and D. K. Sharma, "Searching boolean function using simulated annealing and hill climbing optimization techniques," in *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, pp. 62–64, May 2016.

[23] P. Ganesan, V. Kalist, and B. S. Sathish, "Histogram based hill climbing optimization for the segmentation of region of interest in satellite

images," in *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*, pp. 1–5, Feb 2016.

[24] V. Ravindran and J. Sutaria, "Implementation in arm microcontroller to maximize the power output of solar panel using hill climbing algorithm," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pp. 1234–1240, March 2016.

[25] F. Lan, L. Tao, J. Li, and Z. Yao, "An improved variable step hill-climbing searching algorithm for tracking maximum power point of wind power system," in *2016 China International Conference on Electricity Distribution (CICED)*, pp. 1–6, Aug 2016.

[26] K. KOHARA, T. ISHIKAWA, Y. FUKUHARA, and Y. NAKAMURA, "Stock price prediction using prior knowledge and neural networks," *Intelligent Systems in Accounting, Finance Management*, vol. 6, no. 1, pp. 11–22, 1997.

[27] S. Dobravec, "Predicting sports results using latent features: A case study," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1267–1272, May 2015.

[28] D. Prasetio and D. Harlili, "Predicting football match results with logistic regression," in *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*, pp. 1–5, Aug 2016.

[29] H. R. Patel, S. M. Parikh, and D. N. Darji, "Prediction model for stock market using news based different classification, regression and statistical techniques: (pmsmn)," in *2016 International Conference on ICT in Business Industry Government (ICTBIG)*, pp. 1–5, Nov 2016.

[30] K. Kostoglou, K. P. Michmizos, P. Stathis, D. Sakas, K. S. Nikita, and G. D. Mitsis, "Classification and prediction of clinical improvement in deep brain stimulation from intraoperative microelectrode recordings," *IEEE Transactions on Biomedical Engineering*, vol. 64, pp. 1123–1130, May 2017.

[31] H. S. Kang, C. H. H. Tang, L. K. Quen, A. Steven, and X. Yu, "Prediction on parametric resonance of offshore crane cable for lowering subsea structures," in *2016 IEEE International Conference on Underwater System Technology: Theory and Applications (USYS)*, pp. 165–170, Dec 2016.

[32] K. C. Veluvolu, S. Tatinati, S. M. Hong, and W. T. Ang, "Multistep prediction of physiological tremor for surgical robotics applications," *IEEE Transactions on Biomedical Engineering*, vol. 60, pp. 3074–3082, Nov 2013.

[33] J. F. Engelberger, *Robotics in practice: management and applications of industrial robots*. Springer Science & Business Media, 2012.

[34] J. Luh, "An anatomy of industrial robots and their controls," *IEEE Transactions on Automatic Control*, vol. 28, pp. 133–153, Feb 1983.

[35] C. Lin, P. Chang, and J. Luh, "Formulation and optimization of cubic polynomial joint trajectories for industrial robots," *IEEE Transactions on Automatic Control*, vol. 28, pp. 1066–1074, Dec 1983.

[36] Z. Deng and M. Brady, "Dynamics and control of a wheeled mobile robot," in *1993 American Control Conference*, pp. 1830–1834, June 1993.

[37] T. J. Graettinger and B. H. Krogh, "Evaluation and time-scaling of trajectories for wheeled mobile robots," in *1988 American Control Conference*, pp. 511–516, June 1988.

[38] J. R. Rebula, P. D. Neuhaus, B. V. Bonnlander, M. J. Johnson, and J. E. Pratt, "A controller for the littledog quadruped walking on rough terrain," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 1467–1473, April 2007.

[39] A. Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 932–950, 2013.

[40] M. Hutter, *StarlETH & Co.–Design and Control of Legged Robots with Compliant Actuation*. PhD thesis, ETH Zurich, 2013.

[41] M. Khorram and S. A. A. Moosavian, "Optimal and stable gait planning of a quadruped robot for trotting over uneven terrains," in *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, pp. 121–126, Oct 2015.

[42] S. Nakajima, "Proposal for step-up gait of rt-mover, a four-wheel-type mobile robot for rough terrain with simple leg mechanism," in *2010 IEEE International Conference on Robotics and Biomimetics*, pp. 351–356, Dec 2010.

[43] A. Skaburskytė, M. Luneckas, T. Luneckas, J. Kriaučiūnas, and D. Udris, "Hexapod robot gait stability investigation," in *2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, pp. 1–4, Nov 2016.

[44] K. Kurita, "Estimation technique of friction caused by robot walking motion," in *2013 Second IIAI International Conference on Advanced Applied Informatics*, pp. 268–269, Aug 2013.

[45] R. B. McGhee and A. A. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, pp. 331–351, 1968.

[46] S.-M. Song and K. J. Waldron, *Machines that walk: the adaptive suspension vehicle*. MIT press, 1989.

[47] "Robot Operating System about ros." http://www.ros.org/about-ros/. Accessed: 2017-14-4.

[48] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on Man-Machine Systems*, vol. 10, pp. 47–53, June 1969.

[49] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *The 23rd IEEE Conference on Decision and Control*, pp. 1359–1363, Dec 1984.

[50] E. Oyama, N. Y. Chong, A. Agah, and T. Maeda, "Inverse kinematics learning by modular architecture neural networks with performance prediction networks," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 1, pp. 1006–1012 vol.1, 2001.

[51] R. S. N. Cruz and J. M. I. Zannatha, "Optimal design for a humanoid robot based on passive dynamic walkers and genetic algorithms," in *2015 12th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pp. 1–6, Oct 2015.

[52] Y. Huang, B. Chen, Q. Wang, and L. Wang, "Adding segmented feet to passive dynamic walkers," in *2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 652–657, July 2010.

[53] I. Handžić and K. B. Reed, "Validation of a passive dynamic walker model for human gait analysis," in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 6945–6948, July 2013.

[54] F. Hardarson, "Stability analysis and synthesis of statically balanced walking for quadruped robots," *Royal Institute of Technology*, 2002.

[55] T. Lints, "What is adaptation?," *Info-ja kommunikatsioonitehnoloogia*, pp. 133–135, 2008.

[56] E. L. Ruud, E. Samuelsen, and K. Glette, "Memetic robot control evolution and adaption to reality," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7, Dec 2016.

[57] J. Sparbert, J. Kümpel, and E. P. Hofer, "Motion control of a mobile robot with adaption to the environment," in *2001 European Control Conference (ECC)*, pp. 1145–1150, Sept 2001.

[58] B. Gonsior, S. Sosnowski, M. Buß, D. Wollherr, and K. Kühnlenz, "An emotional adaption approach to increase helpfulness towards a robot," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2429–2436, Oct 2012.

[59] R. J. Crisp and R. N. Turner, *Essential social psychology*. Sage, 2014.

[60] N. Dutt and N. TaheriNejad, "Self-awareness in cyber-physical systems," in *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, pp. 5–6, Jan 2016.

[61] Y. Du, C. W. de Silva, and D. Liu, "A multi-agent hybrid cognitive architecture with self-awareness for homecare robot," in *2014 9th International Conference on Computer Science Education*, pp. 223–228, Aug 2014.

[62] R. Golombek, S. Wrede, M. Hanheide, and M. Heckmann, "Learning a probabilistic self-awareness model for robotic systems," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2745–2750, Oct 2010.

[63] J. H. Holland, "Complex adaptive systems," *Daedalus*, pp. 17–30, 1992.

[64] A. F. Winfield, "Robots with internal models: A route to self-aware and hence safer robots," 2014.

[65] "OpenCV about opencv." http://opencv.org/about.html. Accessed: 2017-17-4.

[66] M. Hutchinson and J. Gallant, "Digital elevation models," *Terrain analysis: principles and applications*, pp. 29–50, 2000.

[67] "Blender about blender." https://www.blender.org/about/. Accessed: 2017-5-3.

[68] T. F. Nygaard, J. Torresen, and K. Glette, "Multi-objective evolution of fast and stable gaits on a physical quadruped robotic platform," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, Dec 2016.