NTNU

Norwegian University of
Science and Technology

A Sensor Calibration and Sensor Fusion Approach with Focus on Tactile Force Estimation for Perception-Driven Obstacle-Aided Snake Robot Locomotion

Vetle Bjørnstad Fredriksen

Master of Science in Cybernetics and Robotics
Submission date: June 2018
Supervisor: Øyvind Stavdahl, ITK
Co-supervisor: Filippo Sanfilippo, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

“Get Schwifty!”— Rick Sanchez, Rick and Morty

Summary

Snake robots have been a research topic of interest since Shigeo Hirose developed the first snake robot in 1972. As with many other types of robots, snake robots are interesting because they could be used to perform tasks too difficult or too dangerous for humans. The idea to design robots similar to snakes comes from what is observed in nature. Real snakes can be seen to navigate through narrow spaces, up steep slopes, and over and around obstacles with ease. They can do this because of their ability to shape their bodies to fit the environment, and through contractions and extensions of muscles, move forward. This is one of the abilities the research community tries to mimic with snake robots - the ability to use the properties of the terrain to its own advantage. In areas where wheeled robots might get stuck, a snake robot could potentially move around easily. When research on snake robots reach the point where their agility reaches a similar level as for real snakes, one can imagine a whole range of scenarios where snake robots are applicable. Inspection of a narrow pipe, mapping of a collapsed building, mine sweeping, and search and rescue operations are all tasks that, in the future, could be performed by snake robots. However, before this can happen, snake robots have to be able to traverse a cumbersome terrain. The technique where a robot uses obstacles to its advantage, rather than being hindered by them, is called obstacle-aided locomotion, and is actively being researched. To achieve obstacle-aided locomotion, the snake is dependent on receiving data from various sensors to obtain information about the environment. This thesis demonstrates how perception-driven obstacle-aided locomotion is implemented for a real snake robot, in a controlled laboratory environment. The main contributions of this thesis are

- finalising the calibration of the force detecting sensors, enabling tactile perception
- implementing sensor fusion, combining tactile perception with visual perception
- replicating simulation demos with the real snake robot, demonstrating perception-driven obstacle-aided locomotion
- distributing developed code to be used freely by anyone

The use of sensor fusion between tactile and visual perception is a big part of achieving perception-driven obstacle-aided locomotion. At the current stage, the snake robot is confined to an indoor laboratory environment, and there is a long way to make snake robots useful in real operations. However, the implementation of perception-driven obstacle-aided locomotion in a laboratory setting is one step along the way.

Sammendrag

Slangeroboter har vært et aktivt forskningsfelt helt siden Shigeo Hirose utviklet den første slangeroboten i 1972. Som med mange andre typer roboter er slangeroboter interessante fordi de kan potensielt brukes til å utføre oppgaver for vanskelige eller for farlige for mennesker. Ideen om å designe roboter som ligner på ekte slanger kommer fra observasjoner gjort i naturen. Ekte slanger beveger seg gjerne gjennom trange områder, opp bratte bakker, og over og rundt hindringer. De kan gjøre dette på grunn av deres egenskap til å tilpasse kroppen til omgivelsene. Gjennom kontraksjoner og ekstensjoner av muskler beveger slangen seg forover. Dette er en av egenskapene forskningsmiljøet ønsker å etterligne med slangeroboter - egenskapen til å utnytte omgivelsene til sin egen fordel. I områder hvor roboter med hjul lettere kan sette seg fast er det mulig at slangeroboter kan bevege seg uten store problemer. Når forskningen på slangeroboter kommer så langt at bevegeligheten deres ligner på den man finner hos ekte slanger, kan en se for seg en rekke situasjoner hvor slangeroboter kan benyttes. Inspeksjon av et tynt rør, kartlegging av en bygning som har kollapset, minerydding, og redningstjenester er alle oppgaver som, i fremtiden, kan bli utført av slangeroboter. Før dette kan skje må slangeroboter være i stand til å traversere ulendt terreng. Teknikken hvor en robot bruker hindringer til sin fordel, i stedet for å bli hindret av dem, kalles hinderassistert lokomosjon og er et aktivt forskningsfelt. For å oppnå hinderassistert lokomosjon er slangen avhengig av informasjon om omgivelsene. Informasjonen kommer fra ulike sensorer. Denne oppgaven viser hvordan persepsjonsdrevet hinderassistert lokomosjon er implementert for en ekte slangerobot, i kontrollerte omgivelser. De viktigste bidragene til denne oppgaven er å

- ferdigstille kalibrering av kraftmålende sensorer, som muliggjør taktil persepsjon
- implementere sensor fusion ved å kombinere taktil persepsjon med visuell persepsjon
- replikere demo fra simulering med den ekte slangeroboten, og med det demonstrere persepsjonsdrevet hinderassistert lokomosjon
- distribuere koden som er utviklet så den kan brukes av alle som vil

Bruken av sensor fusion mellom taktil og visuell persepsjon er en viktig del for å oppnå persepsjonsdrevet hinderassistert lokomosjon. Slangeroboten er for øyeblikket begrenset til å operere innendørs i laboratorieomgivelser, og det er en lang vei å gå før slangeroboter kan brukes i ekte situasjoner. Likevel, implementasjonen av persepsjonsdrevet hinderassistert lokomosjon i laboratorieomgivelser er et steg på veien.

Preface

This work is a master thesis at the department of cybernetics and robotics at the Norwegian University of Science and Technology, Trondheim, written during the spring of 2018. The thesis concerns perception-driven obstacle-aided locomotion for snake robots, which is currently a research topic at NTNU. The work done in this thesis is a continuation of my project thesis, written during the fall of 2017. The workload has been twofold, with both practical and theoretical work. The practical part has mostly consisted of programming and conducting experiments. The theoretical work includes studies of multiple papers on snake robots. In addition to this, sensor fusion is researched and implemented. The work has usually been carried out at room B333, commonly referred to as the snake lab, at NTNU, Trondheim. The room contains a stationary computer with Ubuntu 14.04 installed, and a laptop with Windows 7. Ubuntu is a Linux distribution. The Robot Operating System (ROS) framework is installed on the computer with Ubuntu and most of the programming has been done in C++ in the ROS framework. The laptop has LabVIEW installed, which is used to communicate with the real snake robot. LabVIEW is only used as an interface to communicate with the real snake, and to communicate with ROS and MATLAB. LabVIEW is used to manually control the joint angles on the snake for testing purposes, and to monitor the temperature of the joints. The snake robot has been at my disposal for the duration of the thesis. During the work on the project thesis, I received a folder containing MATLAB code used to communicate with LabVIEW, and hence, the snake robot. To my best of knowledge, this code is written for a previous project with the same snake robot, and is reused in this thesis. The parts of the code used is for performing an experiment with the force detecting sensors. This paper assumes that the reader has basic knowledge of control theory.

I would like to thank my supervisor, Filippo Sanfilippo, for his support throughout this project. First of all, he has created the algorithm, the obstacle triplet model, that is used by the snake. He has also made a big contribution to the project by designing the overall system on the software side. He has given me great advice regarding how to perform the experiments and how to present the data. I would also like to thank him for giving me feedback on this paper, and for providing me with relevant articles regarding snake robots.

I would like to thank Øvind Stavdahl for being the main supervisor for the project and Pål Liljebäck for giving me valuable information about the force detecting sensors on the snake.

I would also like to thank Harald Kjørholt, for his companionship and great work ethic at the lab. He helped my during the calibration of the force detecting sensors, by reading of the sensors while they were being calibrated. He also participated during the verification of the force detecting sensors, and cooperated with me to implement sensor fusion. We had a common objective in each of our master thesis', and that was to implement perception-

driven obstacle-aided locomotion for the real snake robot. Because this was an objective for both of us, the experiment verifying that perception-driven obstacle-aided locomotion functioned as it should was performed together.

Table of Contents

List of Tables

List of Figures

Abbreviations

POAL	=	Perception-Driven Obstacle-Aided Locomotion
ROS	=	Robot Operating System
FSR	=	Force sensing resistor

Chapter 1

Introduction

This master thesis investigates how to implement perception-driven obstacle-aided locomotion (POAL) for a real snake robot. Most of the research conducted on snake robots have been focused on locomotion on smooth surfaces, and traditionally, the way to handle obstacles has been to avoid them (Sanfilippo et al., 2017). With POAL, the focus is shifted, and obstacles are to be exploited rather than avoided. This approach is inspired by real, biological snakes (Sanfilippo et al., 2017). When studying snakes in nature, there is especially one ability researchers want to mimic with their robots. That is the ability to exploit the terrain to gain propulsion. More specifically, snakes uses irregularities in the terrain to push its own body forward. This is what makes snakes so versatile, and gives them the ability to traverse cumbersome terrain, filled with obstacles. If snake robots were able to display similar capabilities, there exist a wide variety of tasks that snake robots could perform. One can imagine a collapsed building where a snake robot could search for survivors among the debris, or underwater pipe inspections. If the inside of an oil pipe needs to be inspected, a snake robot could fit inside the pipe, and use the uneven surface to gain propulsion. A snake robot equipped with sufficient tools could clear areas of mines. These tasks share the property that it is undesirable or impossible to send humans in to perform the same job. However, research on snake robots have not reached the point where such tasks can be carried out yet. For a snake robot to perform such tasks, the robot has to be able to move through an uneven terrain with obstacles, and implementing POAL could be one step in the right direction.

To implement POAL, the robot requires information about its environment. In this thesis, both tactile and visual perception is utilised. The visual perception is handled by another student in his master thesis, (Kjørholt, 2018). In short, sixteen cameras will track the snake in real-time, and the position of every joint will be decided by fusing sensor information from the cameras and the joint angle information from the robot. The tactile perception is implemented so that the snake can acquire some information about the contact with obstacles.

1.1 Background

Snake robots, or snake-like robots, usually consists of a number of joints that can rotate about one or more axes. The first robot developed that falls into this category was developed in 1972 (Hirose and Mori, 2004) by Shigeo Hirose at the Tokyo Institute of Technology. The authors of (Hirose and Mori, 2004) states that the snake-like robot can move forward over uneven terrain, even when the ground is not firm, like a sand dune. Since 1972, there has been some development on snake-robot research. Today, there exists different types of snake robots, including underwater snake robots (Kelasidi et al., 2016). One of the reasons snake robots are a topic of research is the wide range of tasks snake robots could perform. Real snakes are characterised by how their body shapes itself to be able to traverse a cumbersome environment. By using its own body to push against unevenness in the terrain, a snake is able to exploit an uneven surface or obstacles, rather than being hindered by it. A snake robot that can mimic this ability will be better equipped to perform different tasks.

During the fall of 2017, the author began the work of implementing perception-driven obstacle-aided locomotion for the snake robot Mamba, shown in figure 3.4. This work is detailed in (Fredriksen, 2017). To achieve POAL, it is desirable to know the direction and the magnitude of the contact forces between the snake and the obstacles. To obtain these measurements, the strain gauges inside each joint needed to be calibrated and tested. One of the largest contributions of the author's project thesis was the design of a table and a load structure, which made it possible to calibrate the strain gauges while the snake was assembled. During the work done in 2017, and this thesis, some weaknesses in the design have been revealed, which are addressed later in this master thesis. Another contribution was to begin calibrating some of the joints, a process that is continued in this master thesis. This calibration process is one of the steps in achieving POAL.

1.2 Objectives

This thesis describes the work related to achieving perception-driven obstacle-aided locomotion with a real snake robot. This includes fusing sensor data with another thesis and integrating the controllers used in the simulated environment with the real snake. In addition the process of calibrating all force sensors is detailed. The main objectives are

- Finalise the calibration process
- Merge tactile sensor data with visual sensor data
- Replicate the simulation demos displaying perception-driven obstacle-aided locomotion

1.2.1 Tactile perception: an efficient calibration method for strain gauge sensors

This is a continuation of the author's project thesis. It consists of calibrating the strain gauges present in each joint and verifying that they work, using the table and load structure designed during the fall of 2017. Suggestions to further improve this process are included here.

1.2.2 Multi-sensor fusion: merging tactile and visual perception

This task is a collaboration between the author's work and another student's master thesis, (Kjørholt, 2018). This thesis is concerned with the strain gauges on the snake, and the calibration of these to obtain an estimation of contact forces working on the snake. This is the tactile sensor data. The visual sensor data is coming from a camera system present at the lab, and the work is detailed in (Kjørholt, 2018). There are 16 cameras in total, which track the snake in real-time. The visual data from the cameras will be merged with tactile data from the strain gauges.

1.2.3 POAL from simulation to real implementation

This is the final objective. The simulation discussed is a scenario where the simulated snake robot moves forward by using obstacle-aided locomotion. This was achieved by Stian Danielsen, who worked on this project during the spring of 2017. A video of the simulated robot can be found in (Danielsen, 2017). The goal is to make the real snake robot move by obstacle-aided locomotion. More specifically, the *obstacle triplet model* is used to perform the necessary calculations. This algorithm is developed by Filippo Sanfilippo, and is illustrated in figure 1.1. Achieving obstacle-aided locomotion with the real robot comes with some practical challenges. In the simulated environment, one can obtain almost any desired variable, by incorporating the corresponding sensor into the simulation. In the real environment, not only are the available sensors limited, their precision and stability are not as good as in the simulation.

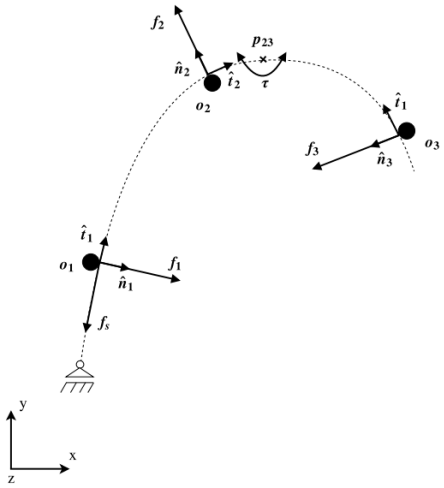


Figure 1.1: An illustration of the obstacle triplet model. The black circles indicate obstacles and the arrows indicate forces. The leftmost obstacle, o_1, is the one closest to the tail. Figure adapted from (Sanfilippo et al., 2016).

1.3 Contributions

To clearly state how this thesis contributes to the project, the main contributions are highlighted below

- Finalise the calibration of the strain gauges in every joint
- Implement sensor fusion with focus on tactile perception
- Implement POAL for the real snake robot
- Distribute the code to be used by anyone

It is noted that the implementation of POAL does not include any further development of the algorithm used to achieve POAL as that is already done by one of the supervisors for this thesis , Filippo Sanfilippo. POAL is also already implemented in a simulated environment, (Danielsen, 2017). Rather, the task consists of implementing the obstacle triplet model for the real snake, using available sensor information.

1.4 Limitations

This master thesis is only concerned with a 2-dimensional environment. That means that the snake is assumed to be planer, and is only able to rotate about the global z-axis, pointing upwards. The robot is also confined to indoor laboratory operations, where the size and shape of every obstacle is known.

1.5 Outline

- **Chapter 1** introduces the project and contains a clear formulation of the objectives. Motivation for researching snake robots is also found here.
- **Chapter 2** presents selected related literature.
- **Chapter 3** describes the different tools and software used in this thesis. The simulated environment and the laboratory environment is also introduced here.
- **Chapter 4** motivates and explains the calibration of the force detecting sensors on the snake.
- **Chapter 5** motivates sensor fusion and explains how sensor fusion is implemented.
- **Chapter 6** contains a mathematical derivation of the obstacle triplet model. The derivation is adapted from (Sanfilippo et al., 2016), and the chapter is copied from the author's project thesis, (Fredriksen, 2017).
- **Chapter 7** contains a detailed explanation of how the implementation of POAL is done.
- **Chapter 8** presents the experiments performed in this thesis.
- **Chapter 9** presents results from the experiments performed.
- **Chapter 10** discusses the results and highlights some sources of error.
- **Chapter 12** contains suggestions for further development of the project.

Chapter 2

Literature review

A lot of research has been done on snake locomotion in general. There exist various approaches to achieve snake-like behaviour in snake robots, but most of the previous research has been focused on movement on smooth surfaces, free of obstacles (Sanfilippo et al., 2017). When operating in an obstacle-free environment, a common gait used is *lateral undulation*. As explained in (Gray, 1946), *serpentine movement*, or *lateral undulation* is a motion where the snake body moves in a sinusoidal pattern, and the whole body follows the head and the neck. As mentioned in (Sanfilippo et al., 2017), lateral undulation is also the fastest gait for snakes. While a lot of research is concerned with smooth environment without obstacles, there is also a lot of research concerned with locomotion in cluttered environments. Real snakes are adept at traversing areas filled with obstacles by exploiting the environment. An implementation of this ability in a snake robot is one of the key features to make snake robots similarly capable of traversing the terrain as real snakes (Sanfilippo et al., 2017). After all, for a snake robot to be of any use, it should be able to move in the presence of obstacles (Liljebäck et al., 2013). There exists three different approaches for handling obstacles, namely

- Obstacle avoidance
- Obstacle accommodation
- Obstacle-aided locomotion

The following section will explain the different approaches and present some of the most recent research on the topic.

2.1 Locomotion in cluttered environments

2.1.1 Obstacle avoidance

Obstacle avoidance is an approach where the robot will avoid contact with obstacles. As stated in (Sanfilippo et al., 2017), a collision may damage the robot, or cause the robot to get stuck. With this in mind, avoiding obstacles makes sense, with regard to both navigation and equipment preservation. A collision could lead to damaged equipment, and this makes obstacle avoidance a reasonable choice. One way to achieve obstacle avoidance is by using artificial potential field (APF) theory. As described in (Lee and Park, 2003), APF works by introducing repulsive and attractive potential fields, surrounding obstacles that should be avoided and the goal to be reached, respectively. Further, the paper mentions a common problem with this approach. The technique uses a gradient descent search to calculate the optimal direction, but with information only about local minimums, the robot may get stuck in a local minimum. The authors claim to have solved this problem by introducing virtual obstacles around these local minimums, enabling the robot to escape, should it get stuck in a local minimum. The robot used in (Lee and Park, 2003) is not a snake robot, but the use of virtual obstacles is still applicable to snake robots. In (Yagnik et al., 2010) a similar approach is used to demonstrate motion planning for a snake robot. APF is used, together with a modified Simulated Annealing algorithm. The Simulated Annealing is what makes the snake robot recover from local minimums.

2.1.2 Obstacle accommodation

Obstacle accommodation is more liberal when it comes to allowing contact with obstacles. With obstacle accommodation, one will allow the robot to make contact with obstacles, but the collisions should occur in a controlled manner to avoid damage to the robot. With this approach, collisions are allowed to happen, and the robot should continue to move in the occurrence of a collision. Compared to obstacle avoidance, obstacle accommodation is advantageous in situations where avoiding obstacles is impossible (Shan and Koren, 1993). A design is proposed in (Shan and Koren, 1993) where a snake robot is able to move through an environment with obstacles. When colliding with obstacles, the number of degrees of freedom for the link in contact is reduced, and the snake pushes itself forward. The authors are able to show that the configuration of the snake can be controlled even when colliding with obstacles.

2.1.3 Obstacle-aided locomotion

Obstacle-aided locomotion differ from the two other approaches in that obstacles are viewed as something that can be exploited, rather than a hindrance. While obstacle avoidance and obstacle accommodation can be useful, they do not quite capture the essence of snake locomotion. Part of the reason why snakes are so agile is their ability to use unevenness in the environment to their advantage. If obstacles are avoided, this is not possible.

Obstacle-aided locomotion is a technique where the robot will use collisions with obstacles as a means to gain propulsion. In (Kano et al., 2011), the authors present a simulation where a snake robot is able to move forward by using obstacle-aided locomotion. Their idea is inspired by how real snakes shape their bodies (to fit the environment) and push against obstacles. In (Gupta, 2007) the author demonstrates obstacle-aided locomotion with a snake-like robot. The robot consists of six joints with force sensing capabilities and is able to move forward by using obstacles as push-points. A push-point is the contact point between the snake and an obstacle, where the snake will exert a force. (Bayraktaroglu et al., 2006) presents a snake-like robot, with ten connected modules capable of rotation about the vertical axis. Similar to this thesis, the authors demonstrate how the robot can move forward by pushing back against obstacles. When their snake is in contact with three obstacles, the snake starts to push against the obstacles and moves forward. If contact should be lost with an obstacle, the head module tries to make contact with a new obstacle. The main result of their research is that their robot reproduces lateral undulation, a way of movement commonly used by snakes in nature. (Sanfilippo et al., 2016) proposes a simplified snake robot model, and a control approach called the *obstacle triplet model*. Their work is meant as a preliminary step towards POAL, and the proposed control approach is the one used in this thesis.

2.2 Environment perception

For a robot to use the environment to its advantage, it is necessary that the robot has some information about the environment. In (Sanfilippo et al., 2017), this task is divided into three categories in the following way

1. *sensing*, using the adequate sensor or sensor combinations to capture information about the environment;
2. *mapping*, which combines and organises the sensing output in order to create a representation that can be exploited for the specific task to be performed by the robot;
3. *localisation*,which estimates the robot's pose in the environment representation according to the sensor inputs.

None of the aforementioned tasks are limited to snake robots, and earlier research does not have to be conducted on snake robots to be relevant. Therefore, the next sections will explore some of the research done on the fields, and the research is not necessarily focused on snake robots.

2.2.1 Sensing

Sensing the environment is a large research topic. First of all, there are many elements of an environment that could be interesting to capture information about. This ranges from temperature, humidity, pressure, and altitude, to what kinds of objects are present, a robot's pose in the environment, and this goes on. Second, there are a high number of different

sensors, and combination of sensors available to capture this information. To narrow it down, this section will focus on the sensing relevant for this project. That would be sensing the contact force coming from obstacles, and finding the magnitude and direction of these forces. In (Taal et al., 2009), the authors use two Position Sensitive Detectors (PSD) in each joint of the snake robot. The sensors are able to capture forces and moments up to 30N and 100Ncm, respectively. Some of the advantages listed for using PSDs to measure forces is that the sensor requires only a short calibration at start-up. Additionally the authors say that the sensor is practically temperature independent and that the quality of the sensor does not decline significantly with time. However, the authors mention that the PSD is less accurate than a general force sensor. This is an example of using an optical sensor to obtain tactile information.

Another way to obtain information about the environment is by using force sensing resistors (FSR). In (Liljebäck et al., 2010) the authors present a snake robot, Kulko, designed with a set of FSRs inside each joint module. The force sensitive resistors are used to measure contact forces applied to the joints. The purpose of the robot Kulko is to demonstrate obstacle-aided locomotion, hence, it is necessary to measure the contact forces applied to the robot. The authors mention that the FSR used is not suited for precision measurements. Still, they assume obstacle-aided locomotion with a snake robot does not require high precision force measurements, justifying the use of the force sensitive resistors. The sensors show a relatively linear relationship between the applied force and the measured conductance. Through an experiment the authors show how the sensors are able to detect which joint a force is applied to, and, to some degree, the magnitude of the force.

2.2.2 Mapping

For a robot to operate autonomously in an unexplored area, the information gathered by the different sensors needs to be combined to create some useful representation of the environment. This includes the location and the shape of obstacles, as well as information about the degree of incline, if that is relevant. Exactly which information is used and how it is represented could vary between applications, but in general, basic information like distance to different objects and shape of close objects is a natural part of the mapping process. (Ohno et al., 2006) uses a 3D camera to capture scans of the environment. The camera contains a light source that emits infrared light that is reflected by objects in the environment. The time it takes for the light to reach the camera after it is reflected is used to measure distance. Based on where the light is emitted from and where the reflected light hits, the camera is able to create a 3D map. The results are best in the lab environment, where the ground is flat. In more realistic scenarios, like a collapsed building, this system did struggle with the 3D map appearing more narrow than the actual scenery.

In (Mobedi and Nejat, 2012), the authors propose a 3D structured light sensory system that can be used to create a 3D map of unknown cluttered environments. The authors demonstrate how their system is able to create a 3D map of an Urban Search and Rescue(USAR) environment. Specifically, the system is able to reconstruct the scene even when shadows, voids, and discontinuities are present. A combination of SIFT techniques and an Iterative Closest Point technique is used to create the 3D mapping from the 3D information. In this

(a) One of the cameras.

(b) The markers tracked by the cameras.

Figure 2.1: This figure shows the different equipment used for the visual perception. The figure is included for completeness, as this thesis is not concerned with this equipment. Rather, this equipment is handled in (Kjørholt, 2018).

project, the only mapping required is the position of the obstacles. The location of the obstacles is acquired by using a camera system present at the lab. Small balls that reflect infrared light is placed on the obstacles and thus the location is obtained. The obstacles are cylindrical with a radius of $10cm$. The camera system accounts for the visual perception in this project. It is an important part of the project, but the author is not involved in the work regarding visual perception. Rather, it is part of another students master thesis, (Kjørholt, 2018). To give the reader an idea, one of the the cameras and two balls are shown in figure 2.1.

2.2.3 Localisation

The localisation problem consists of estimating the position and orientation of the whole snake. There are different approaches that can be used. For example, (Leonard and Durrant-Whyte, 1991) use a single sonar combined with a Kalman filter to predict the pose of their robot in a 2D environment. The authors of (Se et al., 2001) demonstrate the use of a trinocular stereo camera system to obtain both a 3D map of the environment, and the pose of the robot. In other words, mapping and localisation are done simultaneously, and the authors call it SLAMB (Simultaneous Localization And Map Building). To go into more detail, the authors extract SIFT(scale invariant feature transform) features and uses the epipolar constraint and the disparity constraint to match features between the three cameras. From this they are able to build a 3D map of the environment. Another option is the one currently used in this thesis. A camera system tracks the position of joint 1, 5, 9, and 13. This gives the pose of these four joints, and combining the angle between two interconnected joints and simple kinematics, the pose of the robot is obtained. The work done to achieve this is not part of this thesis, but the results are used to achieve POAL.

2.3 Sensor fusion

Sensor fusion is one of the objectives in this thesis, and the scope is limited to the fusion of data coming from tactile and visual sensors. Sensor fusion is an important part of many robotic applications. By utilising sensor fusion, it is possible to obtain more accurate estimates (Lynen et al., 2013) than if no form of sensor fusion is used. The authors of (Lynen et al., 2013) present a framework for multi-sensor fusion extended Kalman filter, which, according to the authors, can improve the robustness of the system to which it is applied. The framework is supposed to be modular and generic, and should be able to handle an unlimited number of sensors.

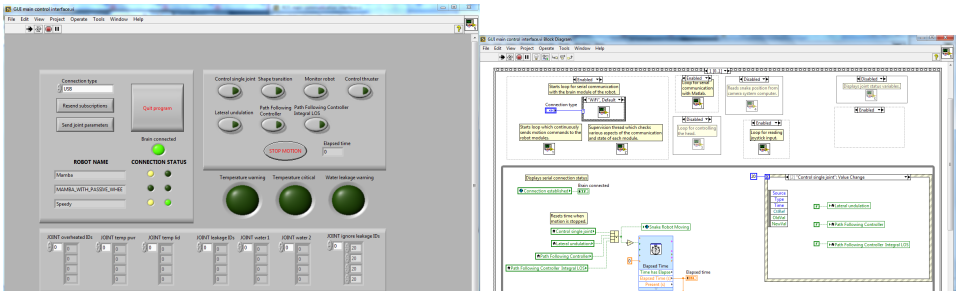
Chapter 3

Tools and Methods

This chapter describes the various tools used in the project. There will be a brief introduction of the different software, and a presentation of the simulated environment. The chapter also introduces the real snake robot.

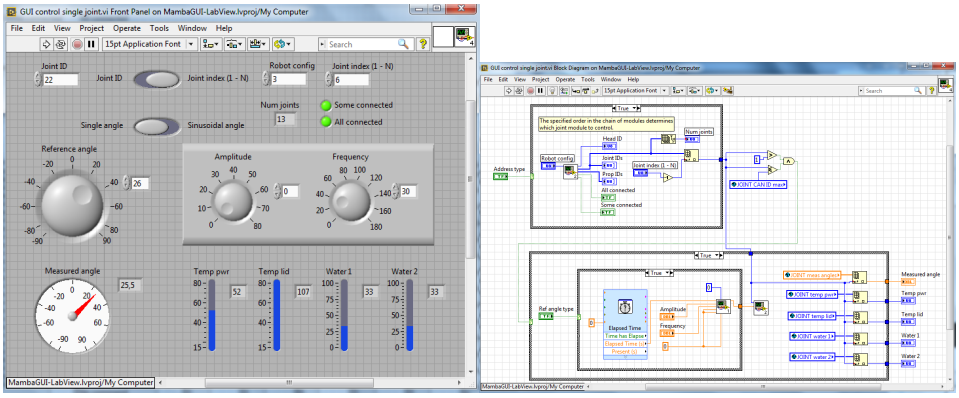
3.1 LabVIEW

LabVIEW is a program developed by National Instruments. It is a language based on data flow, and programming is done in block diagrams. LabVIEW is used in this project to communicate with the real snake robot. Interfacing with hardware is one of the things LabVIEW does really well, and seeing as communicating with the robot is necessary, LabVIEW is well suited for this project. A LabVIEW project is made up of Visual Instruments (VIs), which again is made up of a block diagram and a front panel. The block diagram is where the programming is done, and one can create if-statements, loops, and initialise contact with hardware. The front panel is mainly used as a user interface and for visualising data. Two VI's are shown in figure 3.1, with block diagrams and front panels. The figures are screenshots from the project and illustrate how a small part of the program looks like. LabVIEW is only used as an interface to communicate with the snake in this project. Other tools could be used, but that would require the development of a new interface and low-level programming. As it already existed an interface in LabVIEW, a decision is made to use LabVIEW as the interface. In future iterations of the project, LabVIEW could be replaced by different software, for example ROS.



(a) The front panel of the control interface communicating with the snake.

(b) A part of the block diagram of the control interface.



(c) The front panel of the module that enables the user to control the joint angle of one of the joints.

(d) The block diagram of the module that controls the angle of a single joint.

Figure 3.1: The figure displays screenshots of front panels and block diagrams used in project. The main control interface and the module that controls a single joint are shown.

3.2 Robot Operating System

The Robot Operating System (ROS) is a framework for managing robot applications. ROS offers a wide variety of libraries and device drivers, and can, among other things, handle message parsing. In this project, ROS Indigo is used. Most of ROS's functionality is based on the communication between different *nodes*. In the context of ROS, a node can be seen as a small confined part of the whole program, performing some calculation. To share the result of its own computation with other nodes currently running, a node can *publish* messages to a *topic*. A topic is a channel where data is published by nodes. To acquire the information published to a topic, a node can *subscribe* to said topic. A given topic can have multiple subscribers and publishers. A message published to a topic has to be structured in a certain way. All messages published in ROS are defined in their own .msg file. This file describes what kind of data the message contains, i.e. string, integer, boolean, and so on, in addition to the names of the data fields. The various types of data a message can contain are defined by the ROS community. In addition to simple types like integer and float, a data type can be a composed type. An example is the geometry_msgs/Point type. This message contains three floats, named x, y, z, representing a point in 3D-space. The geometry_msgs prefix means that the Point message belongs to the geometry_msgs package. The user is free to design custom messages consisting of different data types.

One of ROS greatest strengths is the fact that it is open-source. Because of this, solutions found by one person is often distributed so that anyone can use it. This also applies to hardware interfaces, and the official ROS website contains many libraries for interfacing with different hardware. In figure 3.2, a configuration with nodes and subscribers and publishers is shown, illustrating how communication can look like. The figure shows a small part of the whole project to give the reader an idea of how nodes communicate in ROS. The nodes and topics shown are some of the parts involved in achieving obstacle-aided locomotion in the simulated environment. It is noted that there are many more nodes and topics but the ones shown in the figure illustrates communication. While ROS handles a lot of the logic in the project, the actual simulation of the robot is handled by Gazebo, which integrates nicely with ROS.

3.3 Gazebo

Gazebo is a robot simulator and the development is handled by the Open Source Robotics Foundation. Naturally, Gazebo is open-source and is well integrated with ROS. In this project, Gazebo is used to simulate a snake robot, and a number of obstacles present in the scene. In figure 3.3 a screenshot of Gazebo with the snake robot is shown. To make the simulation and the real snake more similar, the number of joints on the simulated snake is increased from 10 to 13. Gazebo can be used to simulate a number of robots in the environment specified by the user. In this project, the scene is just a flat surface with cylindrical obstacles present, but the user is free to design a much more complex environment. Gazebo 2.2 is used in this project, as work prior to this thesis was done in Gazebo 2.2 and upgrading to a more recent version is not the focus of this thesis.

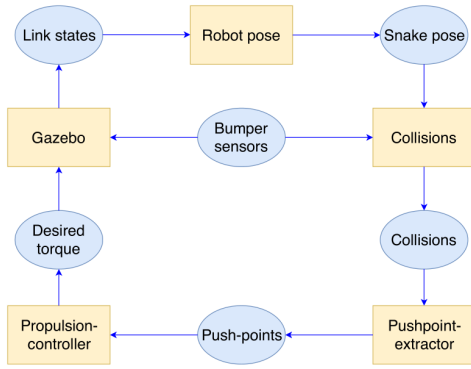


Figure 3.2: A graphical representation of communication between nodes in ROS. The nodes are the rectangles and the topics are the ellipses. An arrow to a topic indicates that the node where the arrow originates publishes data to the topic. An arrow from a topic indicates that the node where the arrow points subscribes to the topic. The nodes and topics shown are part of the simulated environment.

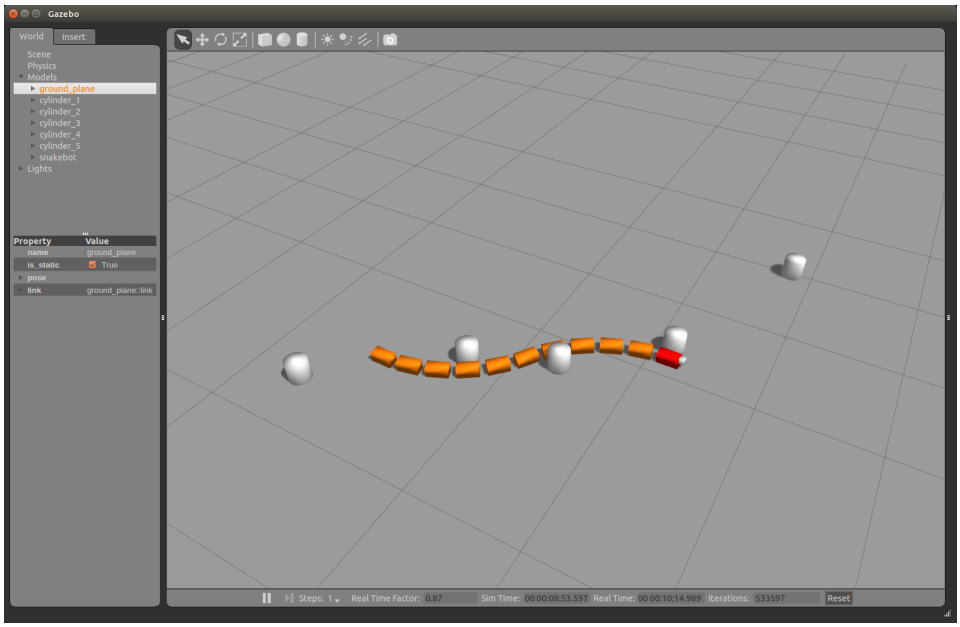


Figure 3.3: A screenshot of Gazebo running with the snake and five obstacles. The head of the snake is coloured red.

3.4 Simulated environment

This project is concerned with both a simulated snake robot and a real snake robot. To give the reader an idea of why this is and how the simulated snake works, this section contains a description of the simulated environment. During 2016 and through the spring of 2017, the goal of this project was to implement POAL in the simulated environment. This work is documented in (Danielsen, 2017). The snake is simulated in Gazebo, introduced earlier, and has been modified to have 13 joints, like the real snake. It is possible to customise everything from the number of joints, and the size of the links, to the weight and colour of the links. All logic concerning the simulated snake is performed by ROS. This includes localisation of push-points and the calculations performed to obtain the desired torque to gain propulsion. In (Danielsen, 2017), the simulated environment is described in more detail, and perception-driven obstacle-aided locomotion is demonstrated. Although the goal of this thesis is to achieve the same with the real snake robot, the simulated environment is still useful. For example, to achieve POAL, contact forces between the snake and obstacles have to be measured. In the simulated environment, the sensors are precise and do not need calibration. In the real world, the strain gauges being used are not as precise and require calibration fairly often. Since the real sensors are less precise than the sensors in the simulation, one will not be able to obtain the same level of precision when measuring contact forces. To investigate whether or not it is possible to achieve POAL with less precise force measurements, the quality of the simulated sensors can be artificially reduced. This is done as an experiment, and is detailed in chapter 8. This is an effective way to investigate the feasibility of achieving POAL with the current snake robot.

3.4.1 Control approach

To be able to move utilising obstacle-aided locomotion, the snake robot uses the *obstacle triplet model*. This is an algorithm developed by Filippo Sanfilippo, and is described more in detail in chapter 6. In short, the obstacle triplet model enables the snake to use obstacles as push-points to gain propulsion. It is required that the snake is in contact with three obstacles simultaneously and one of the joints in contact with an obstacle will push against the obstacle to move the snake forward. In order for this to work, the snake has to know the contact forces applied by each obstacle and the position of the contact. In the simulation, the contact forces are acquired with the use of *bumper* sensors, present in each joint. The contact position can be obtained as the simulated snake always knows where it is and where on the joint a collision occurs.

3.4.2 Controllers

In the simulation there are two ways to control the snake joints, either in position or in torque. Position control is the most simple and can be used to set a joint angle to a desired angle. When the snake is placed in contact with three obstacles, and every joint angle is set to zero, the snake behaves like a spring. That is, the snake tries to straighten out, but

is hindered by the obstacles. This is useful, as the snake needs to stay in contact with the obstacles while performing obstacle-aided locomotion.

The torque controller enables the control of the torque exerted by a joint. The combination of position control and torque control is essential for achieving obstacle-aided locomotion. Position control enables the snake to keep in contact with the obstacles, and torque control enables the snake to push back against one obstacle to gain propulsion. During obstacle-aided locomotion, only one joint is controlled in torque, while the rest are controlled in position.

3.5 Lab environment

The real snake, shown in figure 3.4, is developed at NTNU, and has 13 joints that can rotate about one axis. The snake is wheel-less, but wheels can be mounted on every other joint module. A single joint module, detached from the snake is shown in figure 3.5, with an axis cross showing the body orientation. Every joint has a temperature sensor, an accelerometer, a servo motor, and strain gauges. The joints can rotate $\pm$ 90 degrees. The snake is operated using LabVIEW, introduced earlier. The main objective of this thesis is to achieve perception-driven obstacle-aided locomotion with this snake. To achieve this, a variety of sensors are used to acquire the necessary information about the state of the snake. In short, a combination of tactile and visual perception is utilised to obtain information about the environment. The position of the snake is obtained by using a combination of systems. First, the visual perception system tracking some of the joints sends position data to the computer running ROS. Simultaneously, the computer running LabVIEW sends the measured joint angles to the computer running ROS, and by using simple kinematics, the pose of the entire snake is calculated. The mathematics are detailed in (Kjørholt, 2018).

3.6 Reduced resolution

The sensors available in the simulation are more precise than the sensors available in the lab. Information about contact forces and contact position is important to utilise the obstacle triplet model and to achieve obstacle-aided locomotion. In (Liljebäck et al., 2010), the authors investigate the force sensing capabilities of another snake robot, Kulko, to achieve obstacle-aided locomotion. Kulko uses FSR rather than strain gauges to measure contact force. "The authors conjecture that obstacle-aided locomotion with a snake robot does not require very precise force measurements" (Liljebäck et al., 2010). According to their research, as long as the snake is able to detect a contact, and to some degree the magnitude of the contact, the sensors should be precise enough for POAL. This is in accordance with another paper on snake robots, (Liljebäck et al., 2014), which concerns the snake robot used in this thesis, Mamba, seen in figure 3.4. According to the authors, the strain gauges are not high-precision sensors, but should be sufficient for locomotion in cluttered environments.

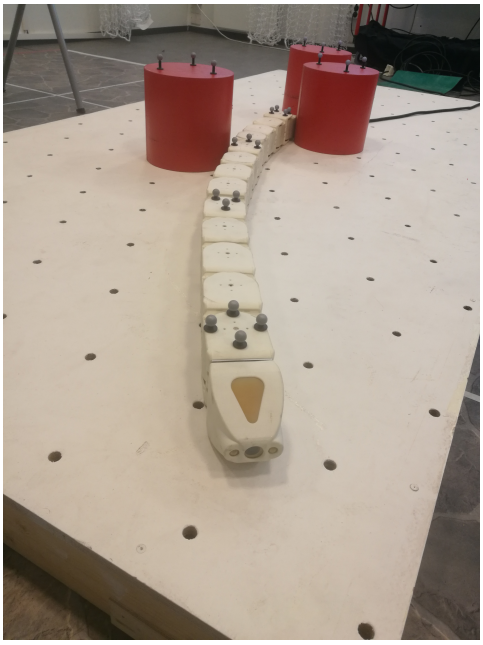


Figure 3.4: Mamba, the snake used in this thesis.

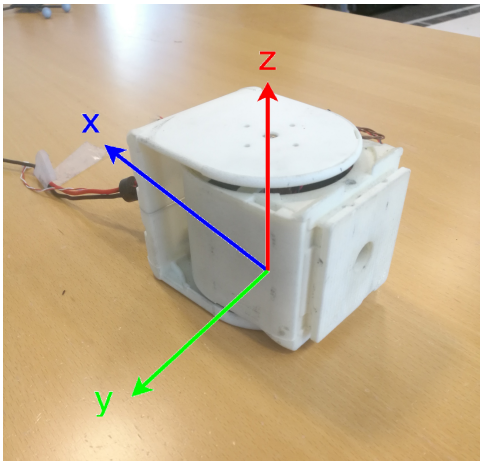


Figure 3.5: A single joint, with an axis cross showing the orientation of the joint's body frame. The x-axis points towards the next joint closer to the head, when the joint angle is zero.

As mentioned, the contact position between the snake and the obstacles are of importance. At the lab, there are some limitations to the precision of the obtained contact position. (Liljebäck et al., 2010) suggests that it is more important to know which joint is in contact with an obstacle, rather than the location of the contact on the joint. To make sure that the claims concerning the feasibility of achieving obstacle-aided locomotion with less precise sensors hold, some changes are made to the simulation. In short, the resolution of the contact forces and positions are reduced, and it is investigated if the simulated snake is still able to perform obstacle-aided locomotion. This experiment is detailed in chapter 8.

3.7 Acquiring sensor data

The control method used to achieve obstacle-aided locomotion, the obstacle triplet model, is explained in chapter 6. To be able to use this method, certain sensor data is necessary. This section will briefly discuss the necessary sensor data, and how the data is acquired. This is summarised in table 3.1.

Table 3.1: The different sensor information necessary.

Required data
Contact forces f
Position of contact c
Snake pose x, y, θ

3.7.1 Contact forces

As described in chapter 6, the contact forces acting on the snake are used in the equations for calculating the torque that should be applied by the snake to gain propulsion. The joints are all fitted with strain gauges, which can, to some degree, detect forces and torques applied to the snake. These strain gauges are not used to obtain the contact forces. The reasoning for this can be found in chapter 10. Instead, the visual perception system is used.

3.7.2 Position of contact

The position of contact between a joint and an obstacle has to be known. Markers tracked by the camera system are placed on top of the obstacles, making the position of obstacles known. When obstacle position and joint pose are known, a simple calculation is used to find the point of contact.

3.7.3 Snake pose

The position and orientation of the snake have to be known. The snake is assumed planar, meaning that only x- and y-coordinates are required for position, and rotation is limited to be about the global z-axis, pointing upwards. The position and orientation are obtained by combining visual data from a camera system, and the measured joint angle between each joint. Using a kinematic chain, the position and orientation of every joint are obtained.

Chapter 4

Tactile perception

This chapter introduces the strain gauges located inside each joint module, including the calibration process of these sensors. The process consists of the calibration itself, and an experiment to verify that the calibration was done correctly, and that the sensors work. The challenges and motivation for conducting the experiment are also found here.

4.1 Introduction

The goal of this thesis is to achieve POAL using tactile perception, amongst other sensor inputs. In the simulated environment, contact sensors are used to obtain all the contact forces working on the snake from the obstacles. It is desirable to use a similar approach for the real snake to be able to reuse most of the logic from previous iterations of the project. As mentioned earlier, the real joint modules are fitted with strain gauges, that in theory, can detect contact forces. Before the strain gauges are calibrated, a lot of the readings simply show a saturated sensor with no load applied. Others seem to be too sensitive when pushed, in that they saturate when only a small push with a finger is exerted. With this behaviour the strain gauges are not feasible for obtaining correct contact forces and therefore require calibration.

In (Liljebäck et al., 2014), an experiment is conducted to calibrate a single joint module, and to investigate the obtained precision of the measured forces and torques. Inspired by this experiment, a new design is proposed in the author's project thesis, (Fredriksen, 2017), to perform the same measurements for every joint module on the snake robot. The reason for a new design is that the load structures used in (Liljebäck et al., 2014), shown in figure 8.3, will not fit around a joint attached to the snake. To perform the experiment in a setting close to an operational scenario, it is desirable to have the snake fully assembled while performing the experiment. A possible solution is a design that makes it possible to conduct the experiment without disassembling the snake. The proposed design facilitates this.

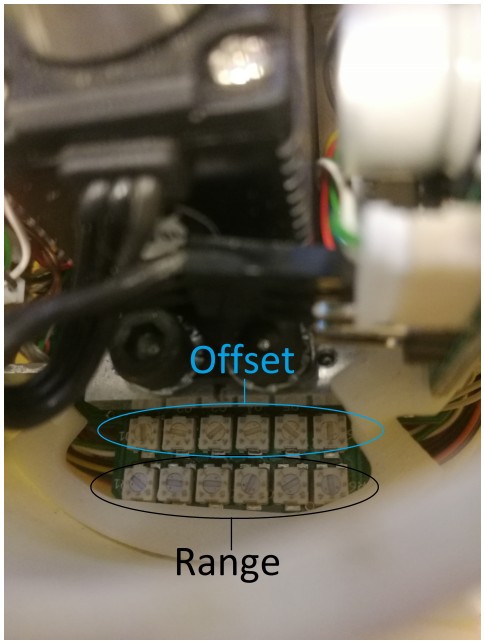


Figure 4.1: The potentiometers inside each joint. They are adjusted by turning them with a tiny screwdriver.

The main challenge regarding this is to design the experiment such that the applied forces along one axis only works in that direction, and at the same time, making the experiment less cumbersome. From the design in (Fredriksen, 2017), some minor changes are made to improve the overall calibration process.

4.1.1 Calibration

Inside every joint, there are strain gauges that measure force along every axis and torque about every axis. Every strain gauge is connected to three potentiometers, controlling the offset, the range, and the gain. The calibration consists of adjusting the potentiometers until the readings are close to 512 when zero load is applied to the joint. Adjusting the offset until 512 is reached is fairly straight forward, as the measurements can be read while adjusting. If the range potentiometer is wrong, adjusting the offset will seem too responsive or unresponsive. To check if the gain is set correctly, some load can be applied, and the measurements should have increased or decreased the expected amount. If only a few weights have been applied, and the reading is 1023, the sensor has saturated, which means that the gain has to be reduced. The inside of the joint is shown in figure 4.1. The potentiometers controlling range and offset are shown, while the ones controlling gain are hidden behind wires and circuit boards. The experiment itself is detailed in chapter 8.

Chapter 5

Sensor fusion with focus on tactile perception

This chapter motivates the need for sensor fusion, and explains how sensor fusion is implemented.

5.1 Introduction

To achieve perception-driven obstacle-aided locomotion, multiple sensor inputs are necessary. The obstacle triplet model, detailed in chapter 6, requires information about the contact forces working on the snake, as well as the pose of the snake. For the snake to obtain the required information, a decision is made to fuse sensor information from a tactile perception system and a visual perception system. As the snake is equipped with force detecting sensors, and the lab is equipped with 16 cameras, these two sensory systems should be sufficient to achieve POAL. The tactile perception system is detailed in chapter 4. The work done with the visual perception system is not part of this thesis and is detailed in (Kjørholt, 2018).

5.2 Motivation

The main reason for implementing sensor fusion as part of this project, is to improve the quality of some of the measurements. More specifically, the idea is to use a combination of the tactile sensors and the cameras to better estimate which joints can be used to gain propulsion. Preliminary tests have revealed that the strain gauges may struggle to correctly indicate contact. Ideally, the strain gauges would clearly show an increase in experienced

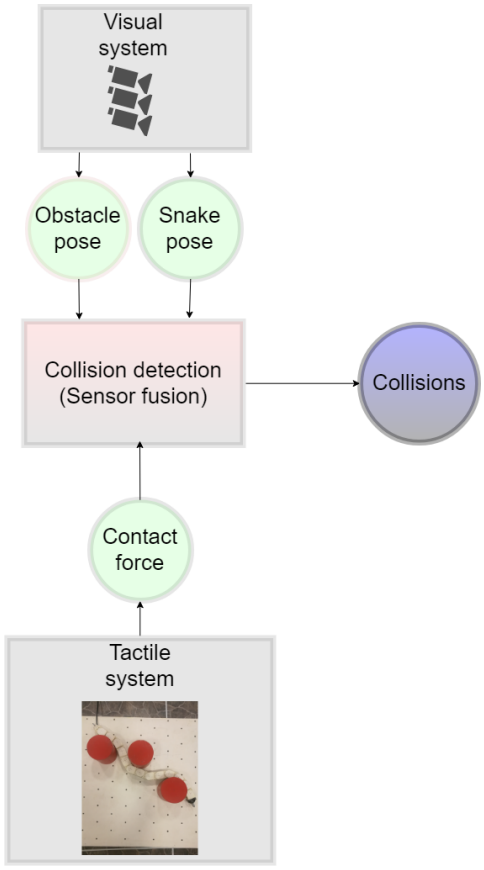


Figure 5.1: A simplified figure showing the sensory system used for sensor fusion, and the result. The result is a list of joints involved in a collision, called Collisions.

force only when contact occurs with an obstacle. Instead, some of the joints indicate high contact with an obstacle even when there is no contact. A reason for why this is happening could be that when the snake is in contact with three obstacles, the whole snake can experience some strain. If it is a deviation between the reference angle and the measured angle for a joint, but the external strain keeps the joint from reaching the reference angle, this could be interpreted as physical contact. The strain gauges also struggle with determining which side of the snake contact occurs on. Knowing the side of contact is essential for the control algorithm to work, hence, there is need for another sensor system in addition to the tactile one.

5.3 Implementation

As briefly explained in the previous section, sensor fusion is used to correctly identify collisions between the snake and the obstacles. A high-level illustration is shown in figure 5.1. As the figure shows, sensor data from the tactile and visual perception systems are combined to find the joints involved in a collision. The tactile perception is the focus of this thesis, while the visual perception is handled in (Kjørholt, 2018). To be clear, the tactile perception system consists of the strain gauges inside each joint module, introduced in chapter 4. Figure 5.1 is simplified, focusing on the actual sensor fusion, which is explained in the following sections.

5.3.1 Visual system

The visual system consists of a camera system and software that enables tracking of the snake and obstacles. This work is detailed in (Kjørholt, 2018) but is included in the figure for completeness. The snake and obstacle poses are sent to the collision detection.

5.3.2 Tactile system

As the snake is fitted with strain gauges, tactile perception is a natural part of detecting collisions. As a part of this thesis, all strain gauges are calibrated and tested. The tactile perception system measures force along every axis of a joint module, but as the obstacles are cylindrical, and the side of the joint module is straight, only the force measured along the y-axis is used to measure contact force. The body orientation of the joint modules can be seen in figure 3.5. The experienced contact is sent to the collision detection.

5.3.3 Collision detection

Collision detection consists of combining tactile and visual data to find the actual collisions. As the strain gauges sometimes indicate a collision when there is none, the visual system is first used to determine which joints could be in contact with an obstacle. This is based on the distance between the joints and obstacles. If the distance is too great, a collision is impossible. Therefore, candidates for collisions are found first. Next, the tactile data is used. Of all the joints that could be in a collision, only the ones with the largest experienced force are selected. The direction of the measured force is not taken into account, so this method is only concerned with the magnitude of the measured force. The reason for this is, as mentioned earlier, that the strain gauges sometimes indicate that contact is coming from the opposite side of what it actually is. By selecting the collisions this way, the set of joints involved in collisions with obstacles is found and the information is passed on to other parts of the system.

Chapter 6

Control model for POAL

This chapter describes the *obstacle triplet model*, which is the algorithm used by the snake to gain propulsion. The mathematical derivation of the algorithm is included, but it is noted that the derivation is not a part of the author's work. The algorithm was developed by this thesis' supervisor, Filippo Sanfilippo, (Sanfilippo et al., 2016), and a former student, Stian Danielsen (Danielsen, 2017). Before the work on this thesis began, the obstacle triplet model is already implemented in the simulated environment. Note that this chapter is copied from the author's project thesis, (Fredriksen, 2017).

6.1 The obstacle triplet model

The obstacle triplet model is presented in (Sanfilippo et al., 2016), and is based on findings from real snakes in nature. (Gray, 1946) found that three contact forces working on the snake have to be present at the same time for the snake to push itself forward. As the snake robot has 13 joints that can rotate in yaw, controlling the snake is a multi-dimensional problem. However, the obstacle triplet model aims to reduce the problem to a one-dimensional one.

6.1.1 Assumptions

In order to utilise this algorithm, some assumptions have to be made. The following assumptions are adapted from (Sanfilippo et al., 2016).

1. a path, S(s), with s being the path length parameter, is known. The obstacle locations, o_1, o_2, o_3, are also known;
2. the snake is always on the path S(s);

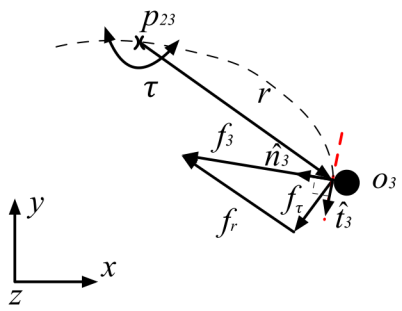


Figure 6.1: Adapted from (Sanfilippo et al., 2016), this figure shows a detailed view of the snake when a torque τ is applied at the point p_{23}. The point is located somewhere on the path between o_2 and o_3.

3. the snake is planar;
4. the snake is continuous;
5. there in no ground or obstacle friction;
6. the snake is at rest;
7. the snake tail link is tethered to the ground, as shown in figure 1.1. The tether is unactuated. No tangential movements are allowed. The tail is not restricted in any other way. A tensile force, f_s, acts along the tangent at o_1;
8. the snake is perfectly rigid except at the point where an internal torque can be applied. The obstacles are perfectly rigid and fixed to the ground surface;
9. we choose to apply an internal torque, τ, at a known point p_{23}, on the path (i.e. snake), between o_2 and o_3, as shown in figure 1.1.

The forces acting on the snake, i.e. f_s, f_1, f_2, f_3 are defined as

$$\begin{aligned} f_s &\triangleq |f_s|\,(-\hat{t_1}), \\ f_1 &\triangleq |f_1|\,\hat{n_1}, \\ f_2 &\triangleq |f_2|\,\hat{n_2}, \\ f_3 &\triangleq |f_3|\,\hat{n_3}. \end{aligned} \tag{6.1}$$

The torque applied at the point p_{23} will produce a counter force, f_τ, shown in detail in figure 6.1. As stated in (Sanfilippo et al., 2016), since it is assumed to be no friction, the total force from o_3, the third obstacle, acting on the snake will be perpendicular to the tangent at o_3.

"With respect to the global reference frame, the torque τ, and the corresponding counter

force f_τ can be denoted as follows:" (Sanfilippo et al., 2016)

$$\tau = \begin{bmatrix} 0 \\ 0 \\ \tau_z \end{bmatrix}, f_\tau = \begin{bmatrix} f_{\tau_x} \\ f_{\tau_y} \\ 0 \end{bmatrix}. \tag{6.2}$$

The contact force f_3 is normal to the snake body at o_3 (figure 6.1) which means that

$$f_3 \cdot \hat{t_3} = 0 \tag{6.3}$$

Looking at the triangle in figure 6.1, one can see that

$$f_3 = f_\tau + f_r, \tag{6.4}$$

because of rules for vector sum. f_τ can be expressed as

$$f_\tau = r \times \tau, \tag{6.5}$$

where τ is the internal torque applied, and r is the distance from o_3 to the point p_{23}. f_r is by definion given by

$$f_r \triangleq |f_r| \frac{r}{|r|}. \tag{6.6}$$

This can be combined to

$$f_3 = r \times \tau + |f_r| \frac{r}{|r|}, \tag{6.7}$$

which can be rewritten as

$$f_3 = r \times \tau + \left[\frac{(\tau \times r) \cdot \hat{t_3}}{\frac{r}{|r|} \cdot \hat{t_3}} \right] \frac{r}{|r|}. \tag{6.8}$$

Following the assumption that the snake is at rest, the following force balance equation must hold

$$f_s + f_1 + f_2 + f_3 = 0. \tag{6.9}$$

In equation 6.9 f_s, f_1, and f_2 are unknown. f_3 is known from equation 6.8. Since the snake is assumed to only operate in a two-dimensional environment, forces acting in the z-direction are not considered, thus making equation 6.9 only relevant for the x- and y-components. Because of the number of unknowns another equation is necessary. One can use the torques exerted on the robot about the global origin by the external forces

$$o_1 \times (f_s + f_1) + o_2 \times f_2 + o_3 \times f_3 = 0. \tag{6.10}$$

Again, since the snake is assumed to only operate in a two-dimensional environment, equation 6.10 is only relevant for torques about the z-axis. Equations 6.8, 6.9, and 6.10 can be combined to get three equations and three unknowns, making the system completely determined. Like mentioned earlier, the whole point of this control approach is to reduce the control problem to being one-dimensional. With the problem now being completely determined, the torque τ can be uniquely computed. A more detailed derivation can be found in (Sanfilippo et al., 2016). A continued derivation can be found in (Danielsen, 2017).

Chapter 7

Implementation of POAL

This chapter describes the overall system, and the ROS nodes that have been added. It also explains how POAL is implemented for the real snake robot at the lab, including work done in MATLAB. The differences from the implementation in the simulated environment is also addressed here.

7.1 System overview

7.1.1 Added modules

The following will present the different parts added to the system as a part of this thesis. The modules running to achieve POAL is shown in figure 7.1. The modules with a thicker black frame is where this thesis has made contributions. Before the author began the work of implementing POAL for the real snake robot, the state of the system is best described by figure 6.17 in (Danielsen, 2017), where the different ROS nodes and topics are shown. As the focus of the project shifted from the simulated environment to the real snake robot, more of the logic is now handled in MATLAB, rather than in ROS. The current system is best represented in figure 7.1. Everything inside the dashed red line is either a ROS node or a ROS topic. The parts outside the dashed line is either hardware or software other than ROS. The interface between ROS and LabVIEW is implemented by Stian Danielsen in (Danielsen, 2017), and is not part of this thesis. The interface between LabVIEW and MATLAB was already in place before the work in this thesis began. The interface between OptiTrack and ROS is an already existing ROS node, (Li, 2014). Note that the changes made to the propulsion controller and push-point extractor are minor. The changes are made so the two nodes can acquire data from the real world instead of simulated data. The following sections will explain the different parts of figure 7.1.

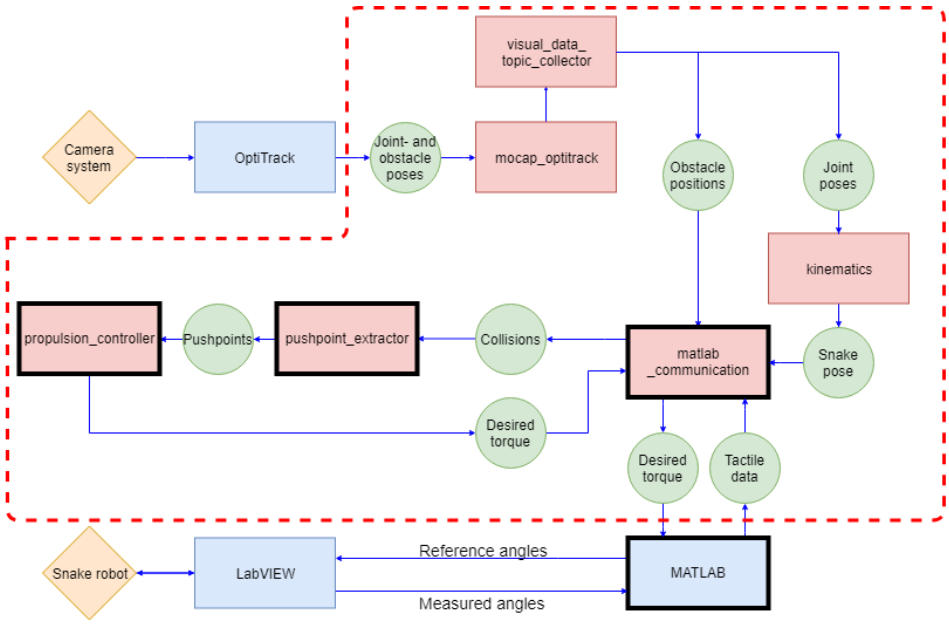


Figure 7.1: Overview of all the parts of the program used to achieve POAL with the real snake. Everything inside the dashed red line is ROS nodes and ROS topics. The green circles are ROS topics, the red rectangles are ROS nodes. The blue rectangles are software running on other computers than the one running ROS. The yellow squares are physical systems. The main focus of this thesis is the modules with a thicker black frame. The other modules are handled in (Kjørholt, 2018).

Tactile data

In chapter 6, it is made clear that the obstacle triplet model requires information about the contact forces working on the snake at the collision points. The joint modules are fitted with strain gauges and this data is transferred to LabVIEW. By using already provided code, all the sensor information from the strain gauges can be obtained in MATLAB. As a part of this thesis, an interface in MATLAB is implemented that publishes data from the strain gauges on a ROS topic, *Tactile data*, in figure 7.1. The reason for not using the already implemented interface between LabVIEW and ROS to transfer tactile data, is that the tactile data is easier to access in MATLAB than in LabVIEW. Because of calibration issues with the strain gauges, addressed in chapter 10, an improvement is done in MATLAB to reduce errors. The first value of each strain gauge is stored at start up, when no load is applied. This value is used as zero for the corresponding strain gauge. This means that if one of the sensors reads 450 at start up, 500 would mean a small force in one direction, and 400 would mean an equal force in the opposite direction. This reduces the need to recalibrate the strain gauges.

MATLAB communication

The ROS node *matlab_communication* is implemented as part of this thesis. The following explains what the node does. By having one node that handles all communication with MATLAB, consistency is ensured. That is how communication with LabVIEW is handled as well. The node subscribes to the topic containing strain gauge data, published by MATLAB. As can be seen in figure 7.1, the node also subscribes to the two topics containing obstacle positions and the pose of the whole snake. The obstacle positions are obtained with the camera system, shown in the top left of figure 7.1. The snake pose is calculated in the *kinematics* node. This work is not part of this thesis, so it should suffice to say that the topic *Snake pose* contains the position and orientation of every joint. This is described in detail in (Kjørholt, 2018). With this information, sensor fusion is used to obtain the collisions between the snake and the obstacles. Details concerning sensor fusion is found in chapter 5. The list of collisions are published to the topic *Collisions*.

From figure 7.1, one can see that the topic *Desired torque* is subscribed to by the MATLAB communication node. This topic contains the torque calculated by the propulsion controller, and which of the thirteen joints should apply the torque. This torque is the τ in figure 1.1.

Push-point extractor

The push-point-extractor was implemented by Stian Danielsen to achieve POAL with the simulated robot. The node subscribes to a topic, Collisions, containing all collisions between the snake and obstacles, and finds three joints to be used as push-points. When moving by obstacle-aided locomotion, the real robot uses the same principles as the simulated robot, so it is desirable to reuse this node to find push-points. This is simply done by

making the node subscribe to the collision topic containing collisions from the real world, rather than collisions in the simulated environment.

Position controller

As mentioned the desired torque is obtained by MATLAB from the *Desired torque* topic. When the simulated robot is moving with obstacle-aided locomotion, the desired torque is sent to a torque controller, and the torque is applied by the selected joint. For the real snake robot, two different controllers are implemented in MATLAB to achieve POAL. The two controllers are not used at the same time. Instead they are both tested separately to investigate which of them gives the best performance. One is a position controller that converts the desired torque to an angle. This angle is used as a reference angle for the joint indicated in the *Desired torque* topic. The angle is sent to LabVIEW from MATLAB. For all the other joints, the reference angle is set to zero. This helps the snake keep in contact with the obstacles. It is not the best approach, and several other techniques are mentioned in chapter 6.2 in (Danielsen, 2017). However, setting the reference angles to zero is deemed sufficient to achieve POAL, and this is demonstrated in the video *POALSensorFusion.mp4*, found in the *videos* folder. The controller will keep increasing or decreasing the joint angle until the error between the reference angle and the measured angle is deemed small enough. Through trial and error, this value is set to 4 degrees. When the error reaches this threshold, the reference angle for every joint, including the one previously used to gain propulsion, is set to zero. This leads to the snake trying to straighten out, and in doing so, moving forward. By moving forward, the idea is that another joint is chosen for propulsion and the aforementioned is repeated. Safety measures is implemented in MATLAB, restricting the reference angle to be inside the interval $[-80, 80]$ degrees. The maximum joint angle is ± 90 degrees for all the joints. Note that the controller just described is the one used to achieve POAL. The other controller implemented is still on the experimental stage and is explained in the following.

Torque controller

The controller is based on the research in (Khatib et al., 2008). In their work, the authors investigate how to implement torque control techniques on joint position-controlled robots. Their results can be summed up by the following two equations

$$\tau_s = k_v[k_p(q_{desired} - q) - \dot{q}] \tag{7.1}$$

$$q_{desired} = k_p^{-1}(k_v^{-1}\tau_s + \dot{q}) + q. \tag{7.2}$$

τ_s is the desired torque, while k_p and k_v are the position gain and velocity gain. q is the joint angle. As the obstacle triplet model is used to calculate a torque, it is desirable to implement a torque controller instead of a position controller. For this controller to work properly, some changes are made in the program. In the simulated environment, during obstacle-aided locomotion, one joint is controlled in torque, while the rest is controlled in position. The desired position for every joint is calculated. This is done in the shape control

module, implemented by Stian Danielsen in (Danielsen, 2017). In short, the position of the three obstacles, plus the position of two extra obstacles, are used to interpolate and find a smooth line. This line defines the desired position of the joints. More details are found in (Danielsen, 2017). The desired torque and desired positions are sent to a Gazebo joint effort controller. That means that the control input for every joint is effort, which is defined by Gazebo. To utilise this functionality for the real snake, the effort calculated is sent to MATLAB, and equation 7.2 is implemented for every joint. However, the results were not as good as when using a position controller. With additional tuning of the gain parameters, it is possible that this controller could work satisfactorily as well. Note that this controller is not used to achieve POAL.

7.1.2 Propulsion controller

As mentioned in section 3.4.1, the snake joints can be controlled either in torque or in position. Position control enables the joint angles to be set to a desired value, while torque control enables the torque about the z-axis to be used as a control input. When every joint is controlled in position, the snake acts like a spring if pushed. A demonstration can be seen in the video attached, called 'SpringModeControl.mp4'. One can see how the ground plane is tilted, causing the snake to slip. When position control is activated, the snake is able to stop the motion by trying to stretch out. When the simulated snake is performing obstacle-aided locomotion, all joints are controlled in position except the one joint where torque is applied. As POAL is already implemented in the simulated environment, it is desirable to reuse most of the code for the purpose of achieving the same for the real snake. Before explaining how this is done, the following section will explain how the propulsion controller works in the simulated environment. Note that the propulsion controller was developed by Stian Danielsen in his master thesis, (Danielsen, 2017).

Inner workings of the propulsion controller

Section 3.2 explains how ROS is based on different nodes publishing and subscribing to certain topics. To calculate the torque the snake should exert to move forward, a number of nodes and topics are in play. The most important ones are shown in figure 3.2. To be clear, there are many more nodes and topics in total, but these are the most important ones, and the ones the author have worked with the most in this thesis.

In the simulation, Gazebo publishes information about the state of all the links continuously to the Link states topic. The node Robot Pose subscribes to this topic and calculates the pose of the whole snake. This information is published to the Snake pose topic, which is subscribed to by the Collisions node. At the same time, when a joint hits an obstacle, the bumper sensors send information to the collisions-node. The message contains information about which joint collided with an obstacle, the position of the contact, and the normal forces resulting from the contact. The collisions-node combines the collision data for every joint, and composes a message published on the collisions topic. This message now contains all the necessary information about the collisions, including on which side

the contact occurs. The push-point-extractor node subscribes to this topic and performs the task of extracting exactly three push-points from the set of collisions. As explained in chapter 6, the obstacle triplet model is based on the assumption that the snake has at least three push-points on alternating sides of its body. The push-point-extractor is able to choose three contact points from the message sent by the collisions-node. The three resulting push-points are sent to the topic *push-points*, which the propulsion controller is subscribed to. The propulsion controller extracts the three push-points, calculates the desired torque, and publishes a message containing the torque and which joint should apply the torque. The simulated snake will exert the torque and hopefully move forward. The message coming from the collisions node, which is received by the push-point-extractor-node has to be constructed in a specific way. The contents of the message is summed up in table 7.1. In order to utilise as much of the previously written code as possible, is is

Table 7.1: The data types with names and descriptions present in the message containing collision information.

Name	Type	Description
link	int32	Number of the link, starting at the tail
contact_side	string	The contact side, left or right
contact_normals	geometry_msgs/Vector3	The normal forces acting on the snake from the collision
contact_tangents	geometry_msgs/Vector3	The tangent forces acting on the snake from the collision
contact_forces	geometry_msgs/Vector3	The actual force from the collision
contact_position	geometry_msgs/Point	The point of the collision, given in the world frame

suggested constructing a similar message, keeping the data fields, but using measured data from the real snake instead of data from the simulation. The following section explains how this is done.

7.1.3 Creating the collision message

Table 7.1 sums up the parts that make up the collision message. Like mentioned earlier, all this information is acquired from the bumper sensors in the simulation. This section will explain how the data is obtained in the real world, and highlight some of the challenges. Some of the information is acquired using sensor fusion, which is explained in chapter 5.

Acquiring collision information

The link number is known from the message sent from LabVIEW to ROS containing joint angles. The contact side is decided by using the pose of the joint and the position of the obstacle with which it collides. In other words, the strain gauges are not used to decide which side the contact is on. In situations where the strain on the snake is small the strain gauges could be used to decide the contact side. This functionality was tested by pushing at the right and left side of each joint, and ensuring that the system read right or left contact. As expected, the system is able to sense the difference between a force

coming from the left or right. However, as the snake tries to straighten out, while being hindered by obstacles, some of the readings contradict what was just discussed. The strain gauge readings from a joint in contact with an obstacle on its right side indicated that the contact was on the left side. As the obstacle triplet model relies on knowing which side contact occurs on, a decision was made to not use the strain gauges for this particular task. During POAL, the strain on the snake is too big for the strain gauges to correctly identify the contact side. A guess for why this error occurred is that when the snake tries to extend, hindered by obstacles, all joints experience some strain. As the joints are connected to each other, the strain on one link can affect the one next to it. This could lead to that the sum of forces on a particular strain gauge wrongly indicate that the force is coming from the opposite than it should. This issue is addressed further in chapter 10.

The contact normals are acquired using the camera system. As all the joint casings are rectangle shaped and the obstacles are cylindrical, the normal force from the contact point will point $\pm$ 90 degrees relative to the joints' orientation. This depends on which side the contact occurs on. The normal force is decomposed into an x- and a y-component, with respect to the global frame. The force is normalised based on the joint's orientation in the global frame.

The contact tangents point in the same direction as the joint, and the pose of all the joint modules are known. The reason for not using the strain gauges to obtain the magnitude of the contact forces is discussed in chapter 10. The contact tangent is decomposed and normalised in the same way as the contact normal.

The contact forces, that is the actual forces working on the snake, are not used in the obstacle triplet model algorithm, so they are not included in the message.

The contact position is obtained using the known position of the joint. As the strain gauges are not able to say anything about where on the joint a collision occurs, the contact position is simply assumed to occur at the centre of the contact side of the joint. As mentioned earlier, this is assumed to be precise enough for POAL, and it has been shown in an experiment to be sufficient. The experiment is detailed in section 8.2. As the global x- and y-coordinate of the contact position is used, the orientation of the joint has to be taken into account. The contact position is acquired using the following equations

$$\begin{aligned} x_{contact} &= x_{joint} \mp \frac{1}{2} w * \sin(\theta_{joint}) \\ y_{contact} &= y_{joint} \pm \frac{1}{2} w * \cos(\theta_{joint}), \end{aligned} \tag{7.3}$$

where w is the width of a joint. The sign of the addition depends on if the contact is on the left or right side, respectively.

Chapter 8

Experiments

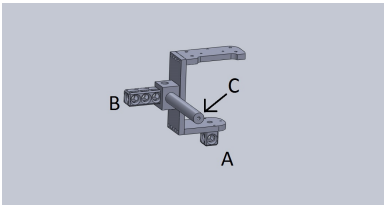
This chapter will describe the different experiments conducted in this thesis. The experiment set-ups are explained here, as well as the purpose of the experiment.

8.1 Tactile perception: calibration process

8.1.1 Experiment set-up

In the author's project thesis, (Fredriksen, 2017), a design is proposed to facilitate the calibration process for every joint. The design includes a table and a structure in aluminium, shown in figure 8.1a and 8.1b, respectively. The calibration process consists of two steps. First, the potentiometers located inside each joint has to be calibrated so that the readings are close to 512 when zero load is applied. This is because the readings range from 0 to 1023. Second, weights are hung from the c-shaped structure, which is fastened around one of the joints. The weights produce an increasing force along one of the joint's axis or a torque about one of the joint's axis. The two designs in figure 8.1 was made to extend a previous experiment performed in (Liljebäck et al., 2014). In their experiment, the load structure shown in figure 8.3 is used to apply the same forces and torques, for a single joint, detached from the rest of the snake. Thus, the new design makes it possible to perform a similar experiment while the snake is assembled. The reasoning behind why the snake should be assembled while performing the experiment is that the snake will be assembled when it is being operated, and the experiment should be performed under similar conditions as when the snake is being operated. In addition to this, the calibration of the strain gauges could be affected by attaching and detaching joints, and by moving the snake. The design in figure 8.1 was first made as a part of the author's project thesis, and is further enhanced here. The table, the load structure, and the weights are shown in more detail in figure 8.2. The small dents running along the table are made deeper so that the load

(a) The table designed for the experiment.



(b) The load structure designed to facilitate the experiment. The points A, B, and C are where the weights are hung from. The 3D-model is made by Daniel Bogen, NTNU.

Figure 8.1

structure does not collide with the table when weights are attached. If the load structure makes contact with the table, the applied load would work directly on the table and not the joint itself. This could lead to errors in the measurements, and could decrease the quality of the calibration process. Another change that is made is to use double-sided tape when applying a force in the y-direction of the joint. The tape is fastened between the table and the joint module. The joint where a force is applied is not fastened, but the joints close to it are. This is done to minimise the forces introduced by the tape.

(a) The designed table, seen from above.

(b) The designed table, seen from the front.

(c) The table seen from the side.

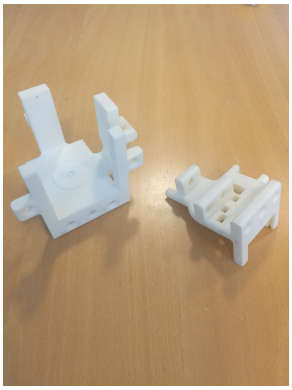
(d) The load structure.

(e) The load structure attached to a joint module.

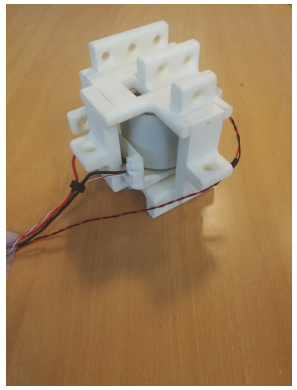
Figure 8.2

(f) A photo of some of the weights used in the experiment, each weighing approximately 250 grams.

Figure 8.2: The different parts used in the experiment. The table is shown from different angles to give the reader a clear understanding of the design. The load structure is shown by itself and attached to one of the joint modules. The weights are also shown for completeness.



(a) The old load structure used in (Liljebäck et al., 2014).



(b) The same load structure attached around a single joint.

Figure 8.3: Two photos showing the load structure both detached, in two parts, and attached to the joint. The two parts fit together and are fastened by small screws.

8.1.2 Experiment

The experiment is carried out by placing the snake on the table. In figure 8.4, one can see that there is one slot in the table for each joint module. The c-structure is slid into place and fastened with two screws at the top. Weights are now hung from either point A, B, or C in figure 8.1b. For all the load cases, the experiment consists of first applying one of the loads, shown in figure 8.2f, weighing approximately 250 grams each. The system reads the strain gauge data for three seconds, then pauses so another weight can be applied. In the cases where force is applied along an axis, twelve weights are used for a total load of 3 kg. For the cases where force is applied in a direction to create a torque about one of the axes, five weights are used for a total of 1.25 kg. The weights are hung either straight down, or over the rail on the table that can be seen in figure 8.1a. When the rail is used, the wire connecting the load to the joint goes straight out from the origin of the joint, so that the force applied is approximately the same as it would be if the load was hanging straight down. The experiment is done six times for each joint. One time for force along each axis, and one time for torque about each axis. In the first three cases, force is only applied along the x-axis, y-axis, and z-axis, respectively. In the last three cases, force is applied in the z-direction to create a torque about the x-axis. Then, force is applied in the z-direction again, but with a different arm to create a torque about the y-axis. Lastly, a force is applied in the y-direction to create a torque about the z-axis. This procedure is summarized in table 8.1. Figure 8.4 demonstrates a part of the experiment.

Table 8.1: The order in which the joint measurements are carried out with corresponding forces and torques.

Order	Measurement	Force	Torque
1	Force x	$[F_x, 0, 0]$	$[0, 0, 0]$
2	Force y	$[0, F_y, 0]$	$[0, 0, 0]$
3	Force z	$[0, 0, F_z]$	$[0, 0, 0]$
4	Torque x	$[0, 0, F_z]$	$[\tau_x, 0, 0]$
5	Torque y	$[0, 0, F_z]$	$[0, \tau_y, 0]$
6	Torque z	$[0, F_y, 0]$	$[0, 0, \tau_z]$

Figure 8.4: A photo taken during an experiment, demonstrating how the weights are hung. In this particular example, the force applied along the joint's y-axis is measured.

8.2 POAL with reduced resolution

As mentioned in chapter 3, claims have been made earlier, stating that obstacle-aided locomotion should be achievable, even without knowing the magnitude of the contact forces, or the exact position of contact. As the strain gauges suffer from some precision errors, these two claims are investigated in the simulated environment. It is also interesting to see if the claims hold, and POAL is achievable with less precise sensors than the ones in the simulated environment. The strain gauges are not capable of detecting where on a joint the contact occurs. If the simulated snake is able to move, using the obstacle triplet model, without knowing the exact contact force or contact position, it is more likely that the real snake can do the same. To investigate this, a simple test is performed in the simulated environment.

8.2.1 Experiment set-up

To utilise the obstacle-triplet model, the snake needs to be in contact with at least three obstacles, on alternating sides of the snake. This is detailed in chapter 6, and illustrated in figure 8.5. This figure illustrates the snake during normal conditions, meaning that the magnitude of the contact forces, and the position of the contact, is known. To investigate how well the snake can perform with less precise measurements, some small changes are done in the code.

Reducing the resolution of contact forces

The contact normals and tangents in figure 8.5, $\mathbf{f_N}$ and $\mathbf{f_T}$, can have any value in $[-1, 1]$. To make the simulation resemble reality, the resolution of these forces are simply reduced, before the propulsion controller calculates the desired torque. Figure 8.6 illustrates this. One can see that the actual measured force $\mathbf{f}$, is reduced to one of four discrete values.

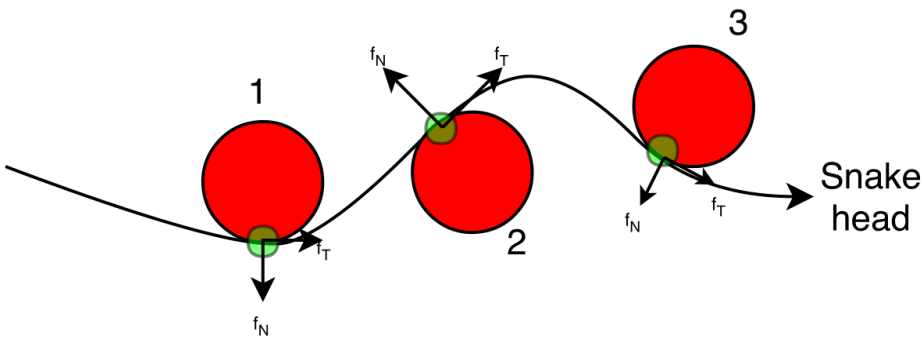


Figure 8.5: An illustration of the snake in contact with three obstacles. The contact points are the green circles and the arrows represent the normal force and the tangent force from the contact point.

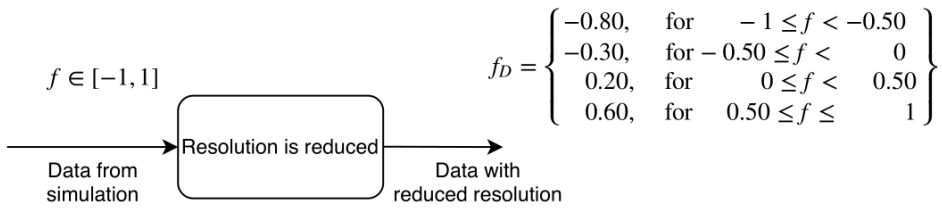


Figure 8.6: A figure illustrating how the resolution of the force sensors are artificially reduced in the simulated environment. f is the force measured, while f_D is the discretized force, which can take one of four discrete values.

The reason for having different values depending on the sign of the force is because of the mathematics done in the propulsion controller. A small code snippet is shown below.

```
double A = det(c1, n1) * (t1.x/n1.x) −
        det(c1, t1) + (det(c2, n2) −
        det(c1, n1)*(n2.x/n1.x))*((t1.y − (n1.y/n1.x)*t1.x)
        /(n2.y − (n1.y/n1.x)*n2.x));

double B = −det(c1, n1) * (n3.x/n1.x) + det(c3, n3) +
    (det(c2, n2)−det(c1, n1)*(n2.x/n1.x))*
    (((n1.y/n1.x)*n3.x − n3.y)/(n2.y − (n1.y/n1.x)*n2.x));
f3 = −fs*(A/B);
```

In the code above, c1, n1, and t1, are the contact position, contact normal, and contact tangent, respectively, belonging to the first contact point, that is, the one closest to the tail. The same goes for contact point 2 and 3. fs is the desired force, specified in ROS. In the calculations above, if some of the forces used to calculate A and B are too similar in magnitude, one ends up with B being equal to, or very close to zero. This in turn makes f3 very large, and f3 is used to calculate the desired torque, meaning that the desired torque also becomes very large. To avoid this, the contact forces are being forced to have different magnitudes. The possible values the contact forces can have are found through trial and error.

Reducing the resolution of contact positions

In figure 8.5, the green circles indicate the position of the contact between the snake and the obstacles, which is given in global x-and y-coordinates. To reduce the resolution of these measurements, the contact is assumed to occur at the centre of the side of any joint in contact with an obstacle. This is done by using the x- and y-position of the centre of the joints, in the global frame, and using trigonometry to find where the contact is assumed to occur. This is illustrated in figure 8.7. The calculations for obtaining the contact position

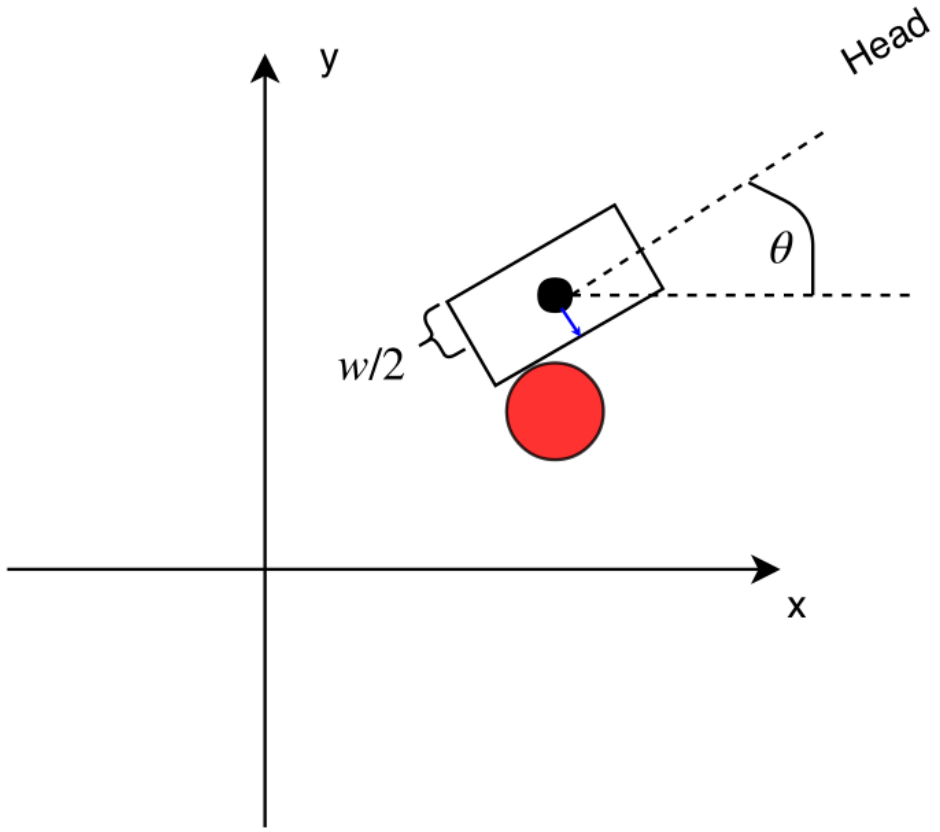


Figure 8.7: An illustration of a joint in contact with an obstacle. The red circle is the obstacle. The blue arrow points to where the contact is assumed to occur. θ is the angle of the joint, with respect to the global x-axis.

are shown here

$$\begin{aligned} x_{contact} &= x_{joint} \mp \frac{1}{2} w * \sin(\theta_{joint}) \\ y_{contact} &= y_{joint} \pm \frac{1}{2} w * \cos(\theta_{joint}), \end{aligned} \tag{8.1}$$

where w is the width of a joint. The sign of the addition depends on if the contact is on the left or right side. Equation 8.1 is the same used to find the contact positions for the real snake.

8.2.2 Experiment

With reduced resolution on both the contact forces and contact position, the simulated snake tries to move by obstacle-aided locomotion. The desired torque for all joints is stored so one can see the performance of the snake. To have something to compare with, another test is performed where the snake is moving by obstacle-aided locomotion under normal conditions. It has already been demonstrated that the simulated snake can move using the obstacle triplet model in (Danielsen, 2017). If the desired torques are similar in the two scenarios, that is a strong indication that the snake can achieve POAL with reduced resolution on the measurements. To demonstrate how the snake performed in these conditions, the desired torques are shown in figure 9.4.

8.3 POAL with the real snake

The main objective of this thesis is to replicate a simulation demo demonstrating POAL. This implies achieving POAL with the real snake. The following describes the set-up at the lab.

8.3.1 Experiment set-up

There is a big difference between a laboratory set-up and an outdoor, realistic scenario. However, the first step in creating something useful is to make it function in a controlled environment. In this thesis, the controlled environment is the laboratory set-up and this is described here, with a photo to give the reader an idea of the conditions in which the robot is operating. The snake is operating on a wooden plate placed on the floor, with dimensions 2.5x1.3 m. The plate is fitted with small circular holes on the surface. The obstacles which the snake will make contact with are circular and are designed to be placed on the plate. The obstacles have two rods sticking out from the bottom, with the same distance between them as the holes on the plate. This ensures that the obstacles easily can be placed at a desired location on the plate, without a chance of the snake moving the obstacles. The plate is kept level during the experiment, as a mentioned limitation of the project is to stay in the 2D-plane, excluding movement in the z-direction. An overhead photo of the set-up is shown in figure 8.8.

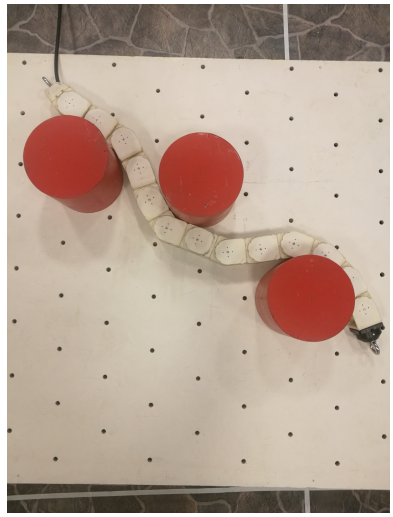


Figure 8.8: A photo of the snake in contact with three obstacles. This is a typical situation where the snake can exploit the obstacles to push itself forward.

8.3.2 Experiment

From the situation in figure 8.8, all the modules in figure 7.1 are used to achieve POAL. The experiments consists of the following tasks

- calculating the pose of every joint module
- obtaining the position of every obstacle
- deciding which joints are most likely in a collision
- calculating the desired torque
- turning one joint to move forward

The pose of the snake is continually calculated as the snake moves. As the obstacles are fastened in the plate, once their position is obtained, it is known for the duration of the experiment. As the snake moves forward, the joint being used to gain propulsion may change, so this is continually being calculated during the experiment. Path planning is outside the scope of this thesis, meaning that the experiment is only concerned with the snake moving forward, without any designated end point. As mentioned in the introduction of this thesis, the goal is to replicate simulation demos demonstrating POAL. Implementing POAL for the real snake is more a proof of concept than anything else. It can be said to be an early step in making snake robots applicable in real scenarios.

Chapter 9

Results

This chapter presents results from the calibration process of the force detecting sensors, as well as results demonstrating the snakes performance during obstacle-aided locomotion, both in simulation and in the lab.

9.1 Tactile perception: calibration process

In this section there are images and tables presenting the obtained results from the experiment. The experiment is performed for all 13 joints, and the results are summed up in tables 9.1 and 9.2. Only the figures from joint 10 are included in this chapter, figures 9.1 and 9.2. Including the plots for all the joints would take up too much space and the payoff is not that rewarding. The rest of the plots can be found as attachments. The figures are generated using code provided by Filippo Sanfilippo, and to the best of the authors knowledge, the code is the same as the one used in (Liljebäck et al., 2014). Hence, the code was not written as a part of this thesis. The figures are shown in the same order as in table 8.1, with measured forces on the left and measured torques on the right. The black line is the actual applied load.

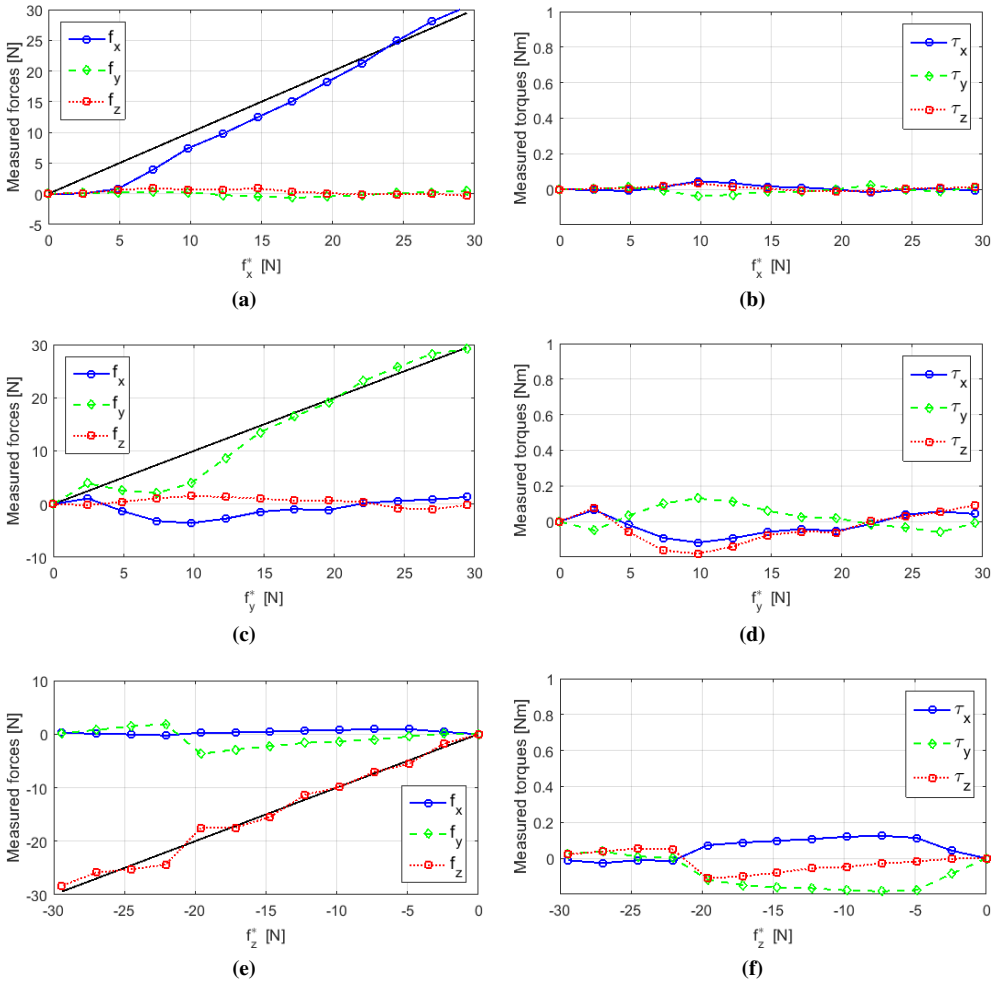


Figure 9.1: Measured forces and torques for the cases where force is applied along one of the axes.

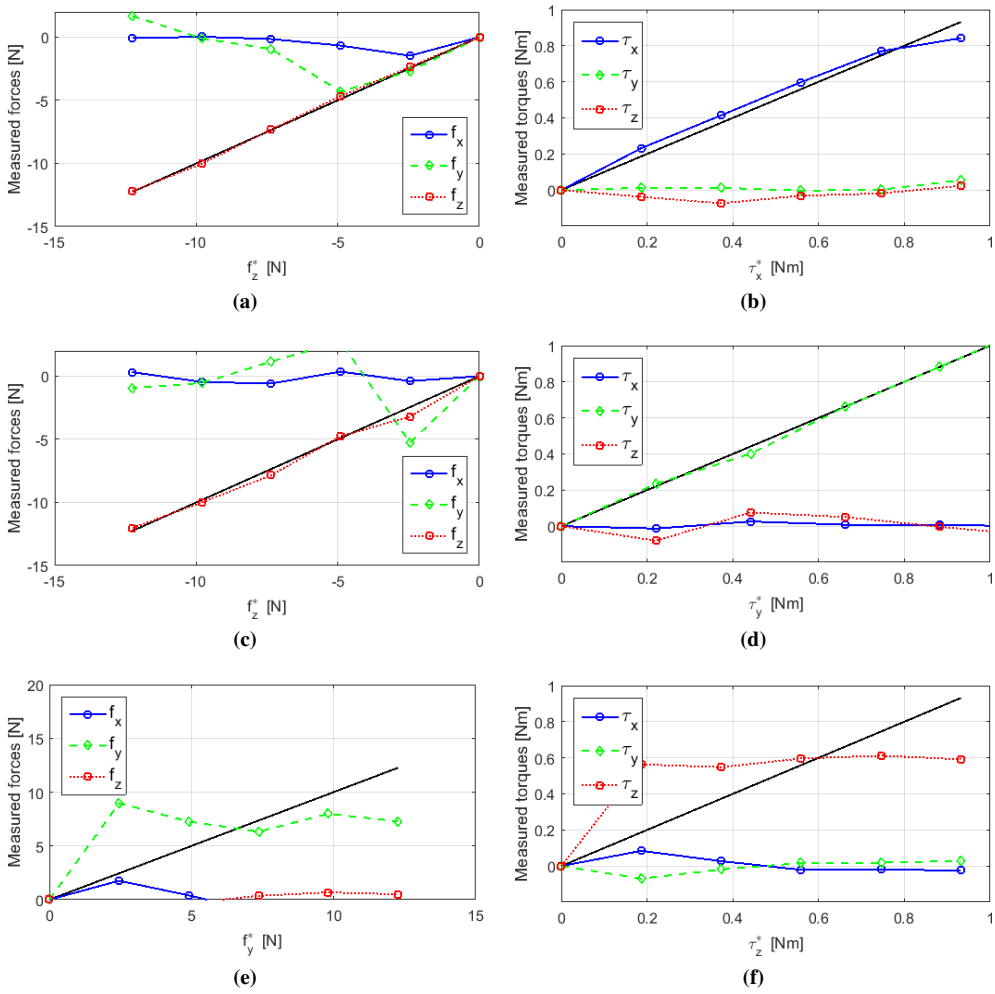


Figure 9.2: Measured forces and torques for the cases where a force is applied to create a torque about one of the axes.

Table 9.1: The RMS errors from the experiment measuring forces and torques.

Joint No.	$\mathbf{f_x}$	$\mathbf{f_y}$	$\mathbf{f_z}$	$\tau_\mathbf{x}$	$\tau_\mathbf{y}$	$\tau_\mathbf{z}$
1	1.20	1.68	2.51	0.09	0.17	0.04
2	2.11	2.23	1.03	0.05	0.13	0.15
3	2.66	1.70	2.91	0.03	0.14	0.13
4	2.05	1.40	2.14	0.09	0.14	0.16
5	1.79	2.11	3.75	0.06	0.13	0.06
6	3.00	1.11	3.77	0.05	0.12	0.11
7	2.17	1.95	3.88	0.06	0.14	0.04
8	1.05	2.54	4.39	0.04	0.10	0.11
9	1.57	2.11	4.61	0.03	0.10	0.08
10	1.41	2.20	0.73	0.05	0.07	0.09
11	1.24	2.81	0.90	0.06	0.06	0.10
12	1.69	0.81	1.37	0.10	0.07	0.11
13	0.29	3.42	1.18	0.08	0.05	0.17

Table 9.2: The MAX errors from the experiment measuring forces and torques.

Joint No.	$\mathbf{f_x}$	$\mathbf{f_y}$	$\mathbf{f_z}$	$\tau_\mathbf{x}$	$\tau_\mathbf{y}$	$\tau_\mathbf{z}$
1	3.10	4.58	7.52	0.35	0.62	0.15
2	5.87	6.34	4.18	0.16	0.58	0.65
3	7.73	8.84	8.60	0.09	0.77	0.56
4	5.96	6.18	6.06	0.29	0.48	0.73
5	4.92	7.84	11.51	0.13	0.39	0.15
6	8.34	4.07	12.37	0.14	0.39	0.37
7	6.13	5.84	11.40	0.16	0.59	0.12
8	3.49	11.16	11.26	0.08	0.30	0.46
9	4.03	6.58	11.89	0.07	0.30	0.26
10	4.09	6.52	2.30	0.13	0.18	0.38
11	3.89	12.23	3.17	0.22	0.19	0.44
12	4.01	3.07	4.37	0.48	0.27	0.45
13	0.84	10.19	3.80	0.40	0.17	0.83

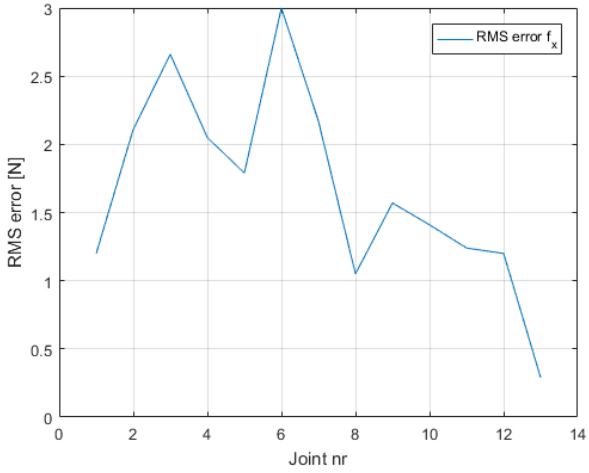
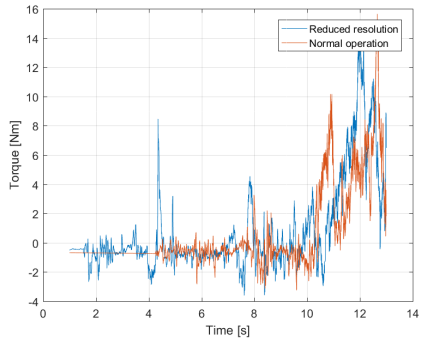


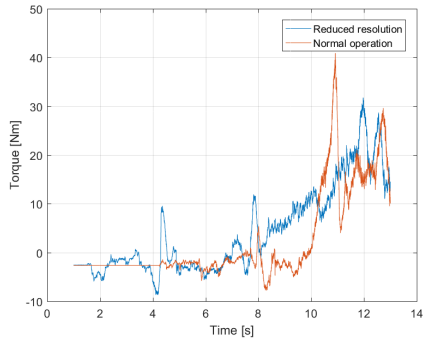
Figure 9.3: The RMS error when applying force along the x-axis, as a function of the joint number.

9.2 POAL with reduced resolution

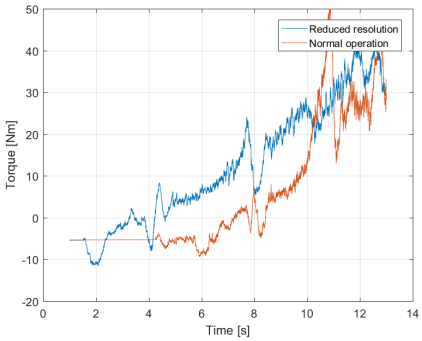
Figure 9.4 shows the desired torque for all thirteen joints with the snake moving by POAL. The blue lines represents the desired torque for the joints in the scenario described, where the resolution of the contact forces and contact points are reduced. The red line represents the torque for the joints in the normal situation. That is when the simulated snake is moving by POAL without reduced resolution on any measurements, which was achieved in (Danielsen, 2017).



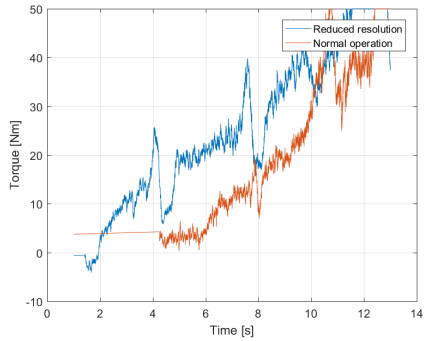
(a) Joint 1



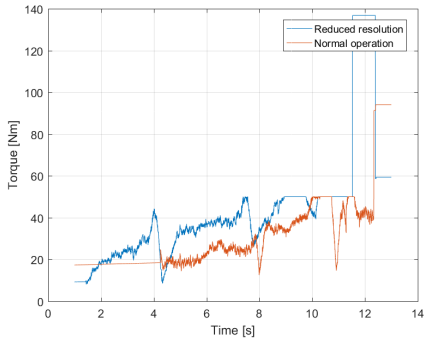
(b) Joint 2



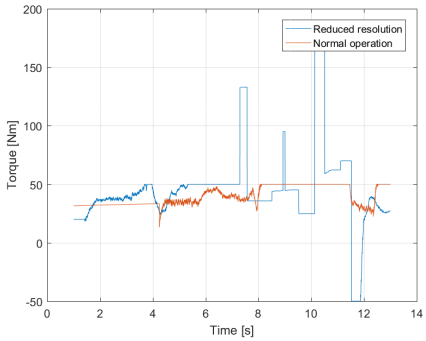
(c) Joint 3



(d) Joint 4

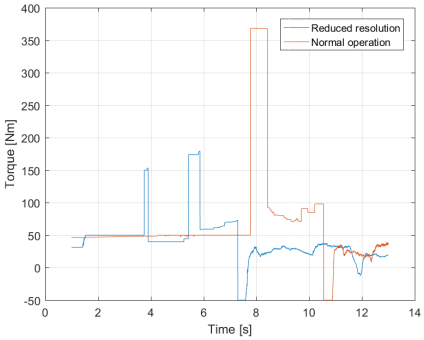


(e) Joint 5

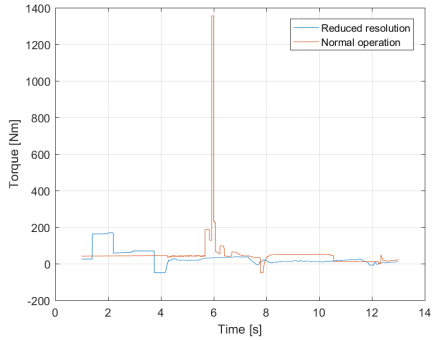


(f) Joint 6

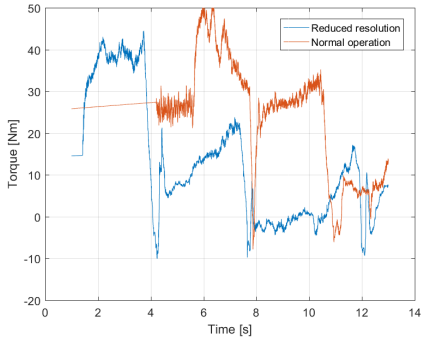
Figure 9.4: Torques for the six first joints, starting at the head.



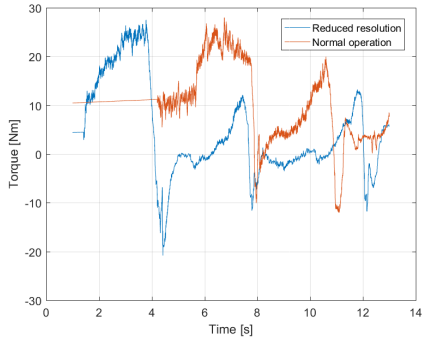
(g) Joint 7



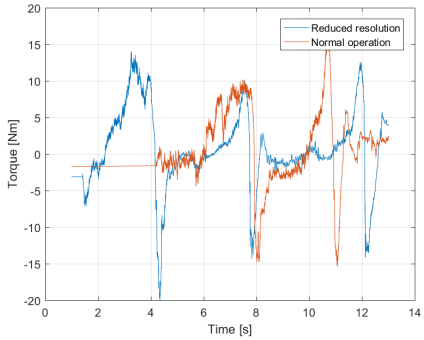
(h) Joint 8



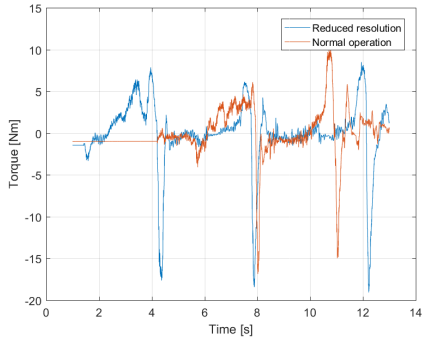
(i) Joint 9



(j) Joint 10

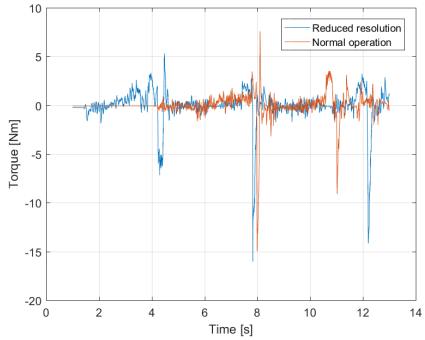


(k) Joint 11



(l) Joint 12

Figure 9.4: Torques for joints 7 to 12.



(m) Joint 13

Figure 9.4: The desired torque for the last joint, the one closest to the tail.

9.3 Replicating simulation demos

The best way to demonstrate POAL with the real robot is with a video of the experiment. The video is attached and is found in the folder *Videos* under the name *POALSensorFusion.mp4*.

Chapter 10

Discussion

This chapter discusses the results presented in chapter 9.

10.1 Tactile perception: calibration process

The results from this experiment is summed up in tables 9.1 and 9.2. For the RMS error, one can see that the largest force errors occur when measuring the applied force in the z-direction. For this particular measurement, the structure in figure 8.1b was not used. As the applied force would be applied only along the thin top of the structure, the weights were instead placed on top of the joint, one at a time. This was done to get a more evenly distributed force. Still, the results indicate large errors for this particular scenario. An explanation for this could be that the joints was poorly calibrated. Especially joints 5, 6, 7, 8, 9, which have MAX errors above 11 N. The experiment was conducted as similar as possible for every joint, which could indicate that the calibration itself was poorly done for the aforementioned joints. As explained in chapter 4, the calibration process consists of adjusting potentiometers controlling range, gain, and offset. This amounts to 18 potentiometers for each joint module. The potentiometers are very sensitive and are adjusted with a screwdriver while checking that the measured value is correct. If this was done only slightly different for some of the joints, the results could differ, although it is uncertain how much. Note that the results were not any better when using the c-structure for the same measurements.

Another result to look at are the forces along the x-axis. To apply force in this case, the weights were fastened to the hook at either the front or the back of the snake. For the six first joints, starting at the head, the weights are fastened at the head. For the remaining joints, the weights are fastened at the tail. Figure 9.3 shows how the RMS error differs for each joint. It seems like the error increases as the joint is closer to the centre of the snake. This could be explained by the fact that the weights are hung from the either the first or

last joint. As the forces applied to the centre joint are measured, the force working in the x-direction is distributed between all the joints from where the load is attached to the joint in question.

It is worth mentioning that the results obtained in this experiment do not represent the actual precision of the strain gauges used on the snake. The already mentioned experiment in (Liljebäck et al., 2014) shows results from a similar experiment, performed on a joint containing the same strain gauges as the ones in this thesis. Their experiment was done on only one joint, detached from the rest of the snake, and the results suggests that the accuracy of the sensors are within $\pm 1\mathbf{N}$ for forces and $\pm 0.04\mathbf{Nm}$ for torques. The results in this thesis suggests that the accuracy lies well within $\pm 5\mathbf{N}$ for forces and $\pm 0.20\mathbf{Nm}$ for torques. Such a discrepancy is probably not caused by poor calibration only. Since so many of the joints in table 9.1 has a worse measurement than the one in (Liljebäck et al., 2014), it is not likely that every joint was poorly calibrated. The most obvious reason for the discrepancy is the design of the experiment. The design shown in figure 8.1b is fastened around one joint at a time, while the snake is assembled. Because of the design, forces and torques are not as well distributed as with the design in (Liljebäck et al., 2014).

10.1.1 Design of table and load structure

An important feature of the design of the load structure was the ability to attach it to any joint whilst having the snake assembled. Therefore, some compromise had to be made with regard to the overall design. Primarily, this lead to the c-shape of the load structure. This design, combined with slots in the table, enables the attaching and detaching of the load structure without moving the snake. However, this also means that the load structure does not wrap around the whole joint, like the one in (Liljebäck et al., 2014). This design decision probably introduces some weaknesses in the design. For example, when applying force along the y-axis, a wire is fastened at the point B in figure 8.1b. The wire is stretched over the rail of the table so that the weights hang straight down. The problem is that the structure does not envelop the joint completely, rather the structure is only wrapped around the top, bottom, and the side which the pull occurs on. This could explain some of the error in the cases were force is applied along the y-axis.

From tables 9.1 and 9.2, one can see that errors for the case $\mathbf{f_y}$ are usually larger than the ones concerning $\mathbf{f_x}$. However, another interesting observation can be made from the two tables. For most of the joints where the $\mathbf{f_x}$ error is larger than the $\mathbf{f_y}$ error, the joint is close to the centre of the snake. This is in accordance with what was mentioned earlier, namely that joints closer to the centre naturally displays a larger error when applying force along the x-axis. As suggested, this happens simply because the load is attached either at the head or the tail of the snake, so the joint in question will experience a smaller force than it should. This is backed up by figure 9.1a, where the measured force is slightly below the applied load, except at the last two measurements.

Another observation can be made from figures 9.2a and 9.2c. In theory, force is only applied in the z-direction in these two cases. The red lines in the figures show that the measured force follows the theoretical line closely. Unfortunately, the green lines, showing

measured force in the y-direction does not stay at zero like it should. In both the figures one can clearly see that the sensors experience a force in the y-direction. This strengthens the case that the design of the experiment is not as good as the one in (Liljebäck et al., 2014). Figures 9.2b and 9.2d show that the measured torque about the x- and y-axis are close to the theoretical values. This indicates that the design at least is able to sufficiently apply the torques in these two cases.

Comparing the approach used in this thesis with the one in (Liljebäck et al., 2014), some of the discrepancies can be explained better. In (Liljebäck et al., 2014), the joint is held in place with a vice, and can thus be rotated so that when measuring force along an axis, that particular axis is pointing downwards. By doing this, the weights can be hung straight down, making the actual applied load closer to the theoretical applied load.

In the new design, a decision was made to keep the snake's pose the same through the experiment. From this it follows that to apply a force in the x- and y-direction, the wire attached to the load should hang over a rail, straight out from the joints origin. At the point where the wire touches the rail, some friction is probably present, and this could explain some of the discrepancies.

Looking at figures 9.1c and 9.1d, which shows the measured forces and torques when force is applied along the y-axis, another observation can be made. At approximately 5 N applied force, the measured force along the y-axis dips below the supposedly applied load. At the same point, the sensor measuring torque about the y-axis seems to experience a torque. At this point the torque should in theory be zero. However, as mentioned earlier, the design of the experiment suffers from some weaknesses that explains some of the deviations.

It is worth mentioning that with the results presented here, showing errors larger than in (Liljebäck et al., 2014), POAL could still be achievable. Firstly because high-precision sensors may not be necessary for achieving POAL. And secondly, the results presented does not represent the actual precision of the strain gauges.

10.1.2 Sensor issues

The strain gauges in the snake suffer from some issues addressed here. For instance, calibration of the sensors is both time consuming and prone to errors. As mentioned earlier, each strain gauge is connected to three potentiometers. The potentiometers are adjusted by inserting a small screwdriver and turning slowly. The potentiometers are sensitive, so only small adjustments are required. As the resolution of the sensors goes from 0 to 1023, each strain gauge rarely holds still at any value. Ideally, one would want the reading at rest at 512 when no load is applied. More realistically, the reading varies slightly, even with no load. Because of this, it is hard to obtain steady readings at 512 with no load. However, that is not a necessity for obtaining good measurements. A more serious issue is that the calibration of the sensors seem to deteriorate over time. Leaving the snake still for a day, after calibrating every sensor makes the sensors drift, thus requiring further calibration. This is a problem, especially since the calibration itself is quite cumbersome.

Another issue that appeared is how the behaviour of one joint can interfere with the strain gauge readings of another joint. This became apparent during preliminary tests to investigate if the snake is able to determine which side the contact with obstacles occurs on. To initiate contact and make the snake behave like a spring, the reference angles for every joint was set to 0. This results in a lot of strain on the snake as it tries to straighten out but is stopped by the obstacles fastened to the ground. This lead to strain gauge readings indicating that the contact occurred on the opposite side of which it actually did. An explanation could be that when a joint tries to reach an angle of 0 degrees, but is physically hindered, nearby joints can perceive the strain as contact from a side.

Visual perception to obtain contact forces

As briefly explained in 7.1.3, the visual perception system is used to acquire the contact normals and tangents, rather than the tactile perception system. The following tries to justify that decision. Through experiments, it is shown that the tactile perception system, the strain gauges, can measure forces and torques relatively precise. Furthermore, as conjectured in (Liljebäck et al., 2010), the precision of the measured contact forces are not the most important for achieving obstacle-aided locomotion. In addition to this, it has been demonstrated that the obstacle triplet model works with a severely reduced resolution on the force measurements. The limitation of the strain gauges became apparent when testing obstacle-aided locomotion. As explained earlier, a requirement for using the obstacle triplet model is that the snake is in contact with at least three obstacles on alternating sides of the snake body. To ensure this was the case, the robot is placed between three obstacles. For the strain gauges to detect a collision, the reference angles for every joint is set to 0 degrees, so that the snake tries to straighten out, and in so, making contact with every obstacle. The sensor output from the strain gauges was not as expected in this situation. Ideally, the strain gauges belonging to the joints actually in contact with the obstacles would display the highest amount of contact. In reality, the joints close to the one in contact also displayed a similar amount of contact. This poses a challenge as joints not in contact with an obstacle should not detect contact forces. The obstacle triplet model is based on finding three points that can be used as push-points. For this to work, the points chosen has to actually be possible push-points. As joints not in contact with an obstacle may still detect large forces, the approach of using the tactile perception system to obtain contact forces is not used. Instead, the visual perception system, which does not suffer from the same problems as the tactile perception system, is used for this task. This approach has led to the successful demonstration of POAL with the real snake.

10.1.3 Improvements

As mentioned earlier, the table and c-structure in figure 8.1 was designed during the authors project thesis during the fall of 2017. In (Fredriksen, 2017), a section discusses weaknesses in the design and changes is made to correct some of them. One point regarded the case where force is applied along the y-axis. In this case, force is applied normal to the snake's x-direction, causing the snake to slip, and eventually the c-structure

would crash into the table. This probably had an impact on previous experiments, and this is addressed here. To reduce the slippage in the y-direction, double-sided tape is fastened under the joints close to the joint module being tested. Whether or not this had a positive effect is hard to say. The results from the project thesis only consisted of joint 2 and joint 12. The RMS error for force in the y-direction was 4.00 and 0.81, for joint 2 and joint 12, respectively. In table 9.1, one can see that the RMS error for force in the y-direction is 2.23 and 0.81, indicating an improvement for joint 2 but no change for joint 12.

When the experiment was performed during the fall of 2017, only the hook at the head of the snake was used to apply force along the x-axis. As this force is distributed over all the joints, it is natural that the results would get worse for the joints further from the head. To reduce this error, a decision was made to fasten the load to the hook at the tail for the joints closer to the tail. Although the data from 2017 is limited, this decision seems to have had a positive effect. The average RMS error for the force in the x-direction is 1.71, while the RMS error for joint 2 and joint 12, was previously 2.02 and 4.08, respectively. Again, this is a small sample size, so one should not read too much into the results. However, the obstacle triplet model does not require that the error in the measurements to be below 2 N, so the results are nonetheless decent.

10.2 POAL with reduced resolution

As can be seen from most of the thirteen plots in figure 9.4, the overall shape of the torque is similar for the two scenarios. A lot of the deviations that can be seen is simply a difference of magnitude between the two scenarios. In the case where the contact forces are restricted to one of four discrete values, the possible torque that can be calculated will also be restricted.

Another reason for the discrepancies is that the data was acquired during two separate runs. It is unlikely that the initial conditions of the two runs were identical, so the lines should not be expected to overlap completely. In all the thirteen figures, the red line is flat during the first three seconds. This is explained by the fact that the snake did not move during the first three seconds after the data acquisition began.

Figures 9.4g and 9.4h show a huge spike in the desired torque for the snake during normal operation. Especially at the 6 second mark in figure 9.4h, where the desired torque is almost 1400 Nm. This could be caused by the contact forces being too similar. This issue is discussed in section 8.2.1, where it is mentioned that the calculated force can get very large when the contact forces are similar in magnitude. Similar spikes can be seen for the blue lines in figures 9.4e and 9.4f, although the magnitude of the spike is smaller. All these spikes can also be partly explained by the fact that the joint that generates the propulsion torque, changes. As the snake moves forward, the joint generating torque moves and loses contact with the obstacle. When this occurs, another joint will have to take over, and thus, a sudden change in the desired torque is explained. In the simulated environment, it is generally not a problem when situations like these occur, as there is no risk of damaging any equipment. The problem arises if too large torques are sent to the real snake. To

reduce the risk of such situations, security measures are implemented to prevent the snake from changing any joint angles too quickly.

With reduced resolution in both contact force and contact position, the simulated snake is still able to perform obstacle-aided locomotion, which is in accordance with the claims in (Liljebäck et al., 2010, 2014). This does not ensure that the same can be achieved with the real snake, but it does make it more likely.

10.3 Replicating simulation demos

The following will discuss the performance of the real snake during POAL, with respect to the video included.

10.3.1 Localisation of collisions

The current implementation makes use of both tactile and visual perception to locate the joints involved in a collision. The snake is able to correctly identify collisions, based on the distance between a joint and an obstacle, and the strain experienced by the joints close enough to an obstacle. As demonstrated in the video *POALSensorFusion.mp4*, this is sufficient for the real snake to move using obstacle-aided locomotion. However, the method of detecting collisions does come with some drawbacks. In certain situations, the joints selected are not the optimal joints for achieving POAL. This can happen in situations where a joint close enough to an obstacle, is experiencing a lot of strain, but the joint is behind the obstacle, with respect to the forward direction of the snake. If this joint is chosen as the propulsion joint, the snake would move backward instead of forward. The reason this could happen is the simple way collisions are detected. A possible solution could be to calculate the desired torque, and determine the direction of the resulting force on the snake, if said torque is applied. If this direction is undesired, another set of collision points has to be selected. To implement this, there is a chance that the tactile perception system has to be replaced with a higher precision system.

10.3.2 Performance

As can be seen in the video *POALSensorFusion.mp4*, the real snake is moving by pushing against the obstacles. It is clear that the snake does not move very fast, and certainly not as fast as the simulated snake. As mentioned earlier, the desired torque is converted to a desired angle. This angle is used as a reference for one joint, while the rest of the joints have 0 as their reference angles. One can see that the rear half of the snake slides slowly forward as the joint turns. The instant the measured joint angle is inside the threshold set at 4 degrees, the whole snake tries to straighten out and moves forward. As mentioned in section 7.1.1, the approach where the reference angle is set to zero for every joint is not the best approach. As every joint but one has zero as a reference angle, the snake will always

try to straighten out, resulting in constant contact forces working on the snake. This could affect the performance, making the snake move slower.

The snake is quite aggressive when straightening out. This is a result of how the controller is implemented in MATLAB. If a joint module slams hard into an obstacle, there is a higher chance of damaging the equipment, and it could be wise to chose a different approach in the future.

Chapter 11

Conclusion

11.1 Finalise the calibration process

One of the objectives for this thesis was to finalise the calibration process of the force and torque-detecting sensors on the snake. There are six sensors on each joint, summing up to 78 sensors in total. The task consisted of calibrating each of them by adjusting small potentiometers, and applying a known force and torque to verify that the error is not too large. This was done for every joint and the results are presented in this thesis. While a lot of the measurements are good and show only small discrepancies, some of them displays large errors. By improving upon the design of the experiment and spending even more time adjusting the potentiometers, perhaps even better results could be obtained. Nevertheless, the sensors have been shown to be able, to some degree, to detect when forces and torques are applied to the snake. As long as the snake is lying flat and there is not too much strain on the snake, the sensors are able to detect in which direction forces and torques are working and the magnitude of the applied force. In an operational scenario, the performance of the strain gauges are notably worse than during controlled experiments.

11.2 Merge tactile sensor data with visual sensor data

As described in chapters 5 and 7, the visual and tactile sensor data works together to find collisions. A simple form of sensor fusion is implemented. Visual perception data is first used to identify the joints which are most likely to be in a collision with an obstacle. This is based on the distance between joint and obstacle. From here, the tactile sensors decide, based on the magnitude of the contact force, which of the joints that are experiencing the largest contact force. This joint is used as a candidate for a push-point, and the visual

system is again used to obtain more information about the collision. As demonstrated, this is sufficient for achieving POAL in a lab environment.

11.3 Replicate the simulation demos displaying obstacle-aided locomotion

The simulation demo shows the simulated snake moving through the environment by pushing against obstacles. The video attached, *POALSensorFusion.mp4* shows the real snake moving by pushing against the obstacles. There are some differences in the movement, but overall, the simulation demo is replicated with the real snake.

Chapter 12

Future work

This chapter suggest improvements to the already implemented design as well as possible new strategies to further drive the project forward.

12.1 Force and torque calibration

The work done to calibrate and test the strain gauges for every joint began during the fall of 2017, as a part of the author's project thesis. The work is finished in this master thesis and it is the author's recommendation to conclude the work on the current strain gauges here. By that, it is meant that it should not be necessary to conduct another experiment consisting of applying weights. A suggestion is to simply use a different set of sensors to acquire contact forces. As discussed earlier, the current strain gauges suffers from some problems, especially in that they require calibration after some time of not being used. The author conjecture that sensors at least able to determine on which side contact occurs on would help improve the performance of the snake. At the lab the snake is currently able to perform POAL, but in a more realistic scenario, like exploring a collapsed building, which is a suggested use for snake robots, it may not be possible to have a camera system detecting obstacles and contact sides. This is why it is important that the sensors fitted on the snake is able to perform this task. There exist many sensors suited for this task, some of which are mentioned in chapter 2.

12.2 Improving sensor fusion

The degree of sensor fusion performed in this thesis can be said to be of a low level. To make the system more robust against bad measurements, a Kalman filter could be

implemented. Especially, this could be used to estimate the position of the snake, reducing the need for the camera system.

12.3 Torque controller

As the obstacle triplet model calculates a torque for the robot to apply, it would be best if a working torque controller is implemented. A suggested torque-to-position controller is implemented here, but requires additional tuning to work properly.

Bibliography

Bayraktaroglu, Kilicarslan, Kuzucu, 2006. Design and control of biologically inspired wheel-less snake-like robot. In: The first IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics.

Danielsen, 2017. Perception-driven obstacle-aided locomotion for snake robots, linking virtual to real prototypes. Master's thesis, NTNU.

Fredriksen, V., 2017. Visual and tactile perception for obstacle-aided locomotion of snake robots.

Gray, 1946. The mechanism of locomotion in snakes. Journal of Experimental Biology.

Gupta, 2007. Lateral undulation of a snake-like robot. Master's thesis, Massachusetts Institute of Technology.

Hirose, Mori, 2004. Biologically inspired snake-like robots. In: In Proceedings of the 2004 IEEE International Conference on Robotics and Biomimetics.

Kano, Sato, Kobayashi, Ishiguro, 2011. Decentralized control of scaffold-assisted serpentine locomotion that exploits body softness. In: In Proceedings of the IEEE International Conference on Robotics And Automation (ICRA). pp. 5129 – 5134.

Kelasidi, Pettersen, Liljebäck, Gravdahl, 2016. Locomotion efficiency of underwater snake robots with thrusters. In: 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics.

Khatib, Thaulad, Yoshikawa, Park, 2008. Torque-position transformer for task control of position controlled robots. In: IEEE International Conference on Robotics and Automation. pp. 1729 – 1734.

Kjørholt, 2018. A sensor fusion approach with focus on visual sensing for perception-driven obstacle-aided snake robot locomotion. Master's thesis, NTNU.

Lee, Park, 2003. Artificial potential field based path planning for mobile robots using

a virtual obstacle concept. In: In Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics.

Leonard, Durrant-Whyte, 1991. Mobile robot localization by tracking geometric beacons. IEEE Transactions on Robotics and Automation, 376 – 382.

Li, 2014. ROS Mocap Optitrack. `http://wiki.ros.org/mocap_optitrack`, online; accessed: 5-October-2017].

Liljebäck, Pettersen, Stavdahl, 2010. A snake robot with a contact force measurement system for obstacle-aided locomotion. In: In proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA).

Liljebäck, Pettersen, Stavdahl, Gravdahl, 2013. Snake Robots: Modelling, Mechatronics, and Control. Springer.

Liljebäck, Stavdahl, Pettersen, Gravdahl, 2014. Mamba - a waterproof snake robot with tactile sensing. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems.

Lynen, Achtelik, Weiss, Chli, Siegwart, 2013. A robust and modular multi-sensor fusion approach applied to mav navigation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems.

Mobedi, Nejat, 2012. 3-d active sensing in time-critical urban search and rescue missions.

Ohno, Nomura, Tadokoro, 2006. Real-time robot trajectory estimation and 3d map construction using 3d camera. In: Proceedings of the 2006 International Conference on Intelligent Robots and Systems. pp. 5279 – 5285.

Sanfilippo, Azpiazu, Marafioti, Transeth, Stavdahl, Liljebäck, 2017. Perception-driven obstacle-aided locomotion for snake robots: The state of the art, challenges, and possibilities. Applied Sciences, 9.

Sanfilippo, Stavdahl, Marafioti, Transeth, Liljebäck", 2016. Virtual functional segmentation of snake robots for perception-driven obstacle-aided locomotion. In: Proceedings of the 2016 IEEE International Conference on Robotics and Biomimetics.

Se, Lowe, Little, 2001. Vision-based mobile robot localization and mapping using scale-invariant features. In: Proceedings of the 2001 International Conference on Robotics and Automation.

Shan, Koren, 1993. Design and motion planning of a mechanical snake. IEEE Transactions on Systems, Man, and Cybernetics 23, 1091 – 1100.

Taal, Yamada, Hirose, 2009. 3 axial force sensor for a semi-autonomous snake robot. In: 2009 IEEE International Conference on Robotics and Automation.

Yagnik, Ren, Liscano, 2010. Motion planning for multi-link robots using artificial potential fields and modified simulated annealing. In: 2010 IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications.

Appendix

Videos and figures from experiment

The following videos and figures are included as digital attachments for this thesis, but can also be found by using the link `https://1drv.ms/f/s!AiGA4KoEn4tntix6I9bBEt-Y0Px6`. Following the link one will find two videos, in the *videos* folder

- **SpringModeControl.mp4**
- **POALSensorFusion.mp4**

In the folder *figures*, one will find plots for all thirteen joints from the calibration of the tactile sensors.

Code

All the ROS code used in this project can be found at `https://github.com/Heltev/snake`.