A Sensor Fusion Approach with Focus on Visual Sensing for Perception-Driven Obstacle-Aided Snake Robot Locomotion

Harald Gard Halvorssønn Kjørholt

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

Is my assessment accurate? - Rick Sanchez, Rick and Morty

Summary

Developing snake-like robots can be advantageous for numerous reasons. Biological snakes are capable of traversing all types of terrain in a very energy efficient and effective way. They achieve this variety of locomotion tasks by utilizing the terrain's roughness in a beneficial way, by pushing against obstacles they encounter and propelling themselves forwards. Mimicking such biological obstacle aided locomotion (OAL) enables the snake to do reconnaissance in a quick manner, and copying this trait is appealing for a variety of applications. Search and rescue missions, firefighting and in general traversing dangerous or unreachable areas are some of the fields that are being investigated for the use of robotic snakes versus mobile robots with wheels or legs.

A form of sensing equipment is desirable for a robot to effectively achieve such locomotion capabilities. Without any means of gathering data from the outside world it would be a virtually blind and hence ineffective approach. This project aims at combining tactile data, coming from the snake's force sensory system, with visual data from an external visual sensor system, to achieve *Perception-driven Obstacle-Aided Locomotion* (POAL). A sensor fusion approach is adopted to effectively combine tactile and visual data. Sensor fusion gives the ability to confirm the reliability of system behavior by cross-referencing the data. This thesis focuses on the visual perception aspect and the implementation of sensor fusion. The most important contributions to the project are

- implementing the visual perception module to collect and visualize all necessary data for Perception-driven Obstacle-Aided Locomotion (i.e. snake position, collision points, collision forces, collision directions, etc.)
- integration with the tactile perception module (i.e. sensor fusion.)
- performing real experiments with POAL (replicating the simulation results that were previously achieved in [1].)
- maintaining and documenting source code with the aim of releasing the framework as an open-source project

Obstacle aided locomotion by sensor fusion is proven plausible in this thesis through experiments. It is demonstrated in attached videos, see Appendix. System development and experiments conducted are as of now strictly for laboratory purposes and do not necessarily represent real-life scenarios.

The main development platform used in the project is Robot Operating System (ROS), used for robot control. The snake is controlled through a robot interface, and tactile data is sent over connections between ROS and the robot interface. A visual sensor interface is used to gather visual data regarding the snake. Simulating the system is done by taking in both visual and tactile data for visualization.

Sammendrag

Utvikling av roboter som etterligner slanger kan være fordelaktig av flere grunner. Slanger er kapable til å traversere alle typer terreng på en veldig energibesparende og effektiv måte. De gjør dette ved å nytte seg av mikrolende på en smart måte. De dytter imot hindringer de støter på og skaper dermed framdrift. Denne formen for hindringsassistert bevegelse gjør slanger i stand til å bedrive rask oppklaring i et gitt område. Dette er grunnen til at etterligne slangeoppførsel er så attraktivt. Søk og redningsoppdrag, brannslukning og generelt akten å traversere områder som anses som for farlig eller unåelig for mennesker er noen felter som undersøkes med tanke på slangerobotteknologi kontra hjul- eller belte-drevne roboter. En form for sanseorgan må til for at roboten skal kunne interagere med omgivelsene. Dette prosjektet prøver å kombinere taktil data fra slangerobotens interne sensorer, med visuell data fra et eksternt kamerasystem for å oppnå *kamerabasert hindringsassistert bevegelse* (KHB). En sensor fusion tilnærmingsmetode er brukt for å kombinere den taktile og visuelle dataen. Sensor fusion gir et system egenskapen til å sjekke system-validitet ved å kryssreferere informasjon fra ulike sensorer. Denne avhandlingen konsentrerer seg om den visuelle delen av slangens sanseorgan og implementeringen av taktil data for sensor fusion. De viktigste bidragene til prosjektet er

- implementere den visuelle oppfatnings-modulen for å samle all nødvendig data som skal til for å oppnå KHB. (i.e. slangens posisjon, kollisjonspunkter, kollisjonskrefter, kraftretninger, etc)
- integrere visuell oppfatnings-modul med taktil oppfatnings-modul (i.e. sensor fusion)
- utføre reelle eksperimenter (replikere tidligere oppnådde, simulerte resultater)
- opprettholde og dokumentere kildekode i den hensikt å utgi rammeverket som et åpen-kilde prosjekt

KHB ved hjelp av sensor fusion er bevist mulig i denne avhandlingen gjennom testing. Dette er demonstrert i vedlagte videoer, se Appendiks. Utviklingen og testingen av systemet er for nå kun for laboratorie-hensikter og gjenspeiler ikke nødvendigvis virkelige senarioer.

Hovedplattformen for utviklingen er Robot Operating System (ROS), brukt for slangekontroll. Slangen kontrolleres gjennom et robot-grensesnitt og taktil data er sendt over forbindelser mellom ROS og dette robot-grensesnittet. Et grensesnitt for visuell data er brukt for å hente in visuelle data fra slangen. Simuleringsprogrammet tar in alt av data for system-visualisering.

Preface

This thesis is the end work of five years studying Cybernetics and Robotics at the Norwegian University of Science and Technology - spring 2018. It is part of an ongoing, comprehensive project concerning robotic snakes and their serpentine motion through rough terrain. A substantial part of the framework in Robot Operating System (ROS) was created by [2] and [1] prior to project start, with additional logic supplemented through the development of this thesis.

This thesis is not a continuation of my project thesis, written in the fall of 2017. Hence some time was spent comprehending components integrated with the system, together with reading up on relevant literature. Several papers have been provided by my supervisor, Filippo Sanfilippo. The rest is a result of my own literary research.

The project is composed of two parts sharing the same end goal. An investigation from a visual and a tactile perspective. The UAV laboratory has therefore been shared with Vetle B. Fredriksen working on the tactile part of the system.

Two Dell computers running Ubuntu 14.04 and Windows 7 and a Dell laptop running Windows 7 were provided, containing all necessary software platforms. So were the utilized robotic snake (Mamba) and the camera system OptiTrack.

I would like to thank my supervisors Øyvind Stavdahl and Filippo Sanfilippo for the opportunity to investigate the field of snake locomotion, and for all the enlightening talks to ensure progress.

I also want to direct thanks to the mechanical workshop at ITK for giving me the sufficient tools and help.

Special thanks go to Vetle B. Fredriksen who has been a companion throughout the entire period working on the project, and an indispensable asset. He has aided in testing and a portion of the results are achieved in partnership. Achieving Perception-driven Obstacle-Aided Locomotion (POAL) is the end result of a considerable collaborative work.

Harald Kjørholt
Trondheim, June 2018

Table of Contents

List of Tables

List of Figures

Nomenclature

checkpoint joint	=	joint from OptiTrack with known position and orientation, functions as a reference joint
dead-reckoning	=	determine position base on previous position and estimated velocity and course
EKF	=	Extended Kalman Filter
IDE	=	Integrated Development Environment
IMU	=	Inertial Measurement Unit
kinematic chain	=	a constellation of links interconnected by joints to provide motion based on a mathematical model for a mechanical system
middle joint	=	joint right between two checkpoint joints
obstacle accommodation	=	controlled collisions with obstacles
obstacle aiding	=	using obstacles to create motion
obstacle avoidance	=	the act of avoiding obstacles in a path
obstacle triplet model	=	setup consisting of three obstacles for three contact points, reducing the number of system dimensions to one
odometry	=	the use of motion sensor data to estimate changes in position over time
pitch	=	rotation about the y-axis
POAL	=	Perception-driven Obstacle-Aided Locomotion
point cloud	=	a set of data points in space
pose	=	the position and orientation of a body
potentiometer	=	an adjustable voltage divider
pushpoints	=	contact points between the snake and resistance points on a surface
RB	=	Rigid Body
rigid body	=	constellation of markers that hold the same position relative to each other
roll	=	rotation about the x-axis
sensor fusion	=	integrating modules with different types of sensory into one system
strain gauge	=	a device for measuring strain on an object
tactile data	=	data registered by sensors, from physical contact
yaw	=	rotation about the z-axis

Chapter 1

Introduction

1.1 Robotic Snakes

Snakes have the ability to traverse all types of terrain graciously and effectively. They achieve this by using the roughness of the micro terrain to their advantage. They seek out resistance points in the surface with the intent of performing controlled collisions [7], utilizing the obstacles for improved motion efficiency. Researchers have adopted this concept of obstacle aided undulation and over the years a variety of artificial snakes have seen the light of day. The first registered robotic snake to achieve serpentine movement dates back to 1972 with the *Active Cord Mechanism model ACM III*, see figure 1.1 for a picture of this snake robot [4]. Further evolution of the concept has introduced more complex systems and the technology is applied in a number of applications. From search and rescue missions [8] to colonoscopies [9].

The typical snake robot consists of a series of interconnected links. What connects the links together is called a joint. Each joint is equipped with a means of generating work (i.e. servo motor) and sensors that can detect forces acting upon them. A form of dept sensing equipment is usually placed on the snake (e.g. camera, laser, ultrasonic sensor etc.). The sensing device is in charge of gathering information from the surroundings, enabling the snake to react to its environment.

1.2 Background and Motivation

Many aspects of the field of robotic snakes are still in their cradle. Both with respect to an obstacle handling aspect and a data utilization aspect. An example is autonomous, perception aided snakes.

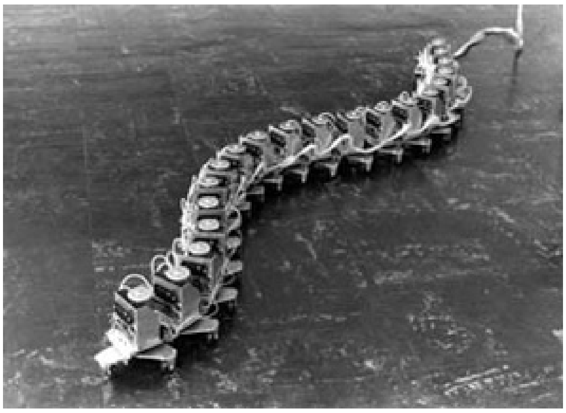


Figure 1.1: The *Active Cord Mechanism model ACM III*, the world's first robotic snake to achieve serpentine movements [4], in 1972.

1.2.1 Obstacle Handling

Obstacle Aided Locomotion (OAL) in robotics is another relatively unexplored territory. Achieving OAL is of interest because it opens up a whole new area of robotics. Snake robots are exceptional in that they are designed to utilize obstacles to create *pushpoints*. Wheeled or legged based robots lack this ability and are often designed for *obstacle avoidance*. Obstacle avoidance is the opposite of OAL. The goal here is to avoid obstacles along the robot's path [10]. Both techniques aim to know their positioning relative to blocking obstacles. They diverge at the point of deciding what to do post discovery.

In [11] a third option regarding handling obstacles is introduced. It is called *obstacle accommodation* and is a more relaxed form of obstacle avoidance. It aims for controlled collisions with obstacles as to not damage the robot. Obstacle accommodation can be a solution when traversing cluttered environments without the intention of utilizing the obstacles, but where collisions are imminent.

A robotic snake, or any other traversing entity, may be better off by traversing the planar ground as a default way of motion, not seeking obstacles unless it is necessary for advancing forward. That is, using obstacle aided locomotion as a supplement rather than a replacement for conventional motion.

1.2.2 Sensor Fusion

Giving a robot the ability to sense the outside world increases its potential for autonomous behavior, where the snake can react to the environment without being controlled remotely. Equipping the robotic serpent with different types of sensors is advantageous since it picks up a greater spectrum of data, making it possible to cross-check data of multiple types, thus creating a more reliable system. Combining sensors in this manner is called *sensor fusion*. See section 6.1 for a more detailed explanation.

1.3 System Concept Overview

This thesis investigates the possibility of combining optical surveillance of a snake, with the tactile data registered from the snake's internal sensory. The purpose is to develop a sensor fusion based system to achieve *Perception-driven Obstacle-Aided Locomotion* (POAL). The surveillance system is a configuration of cameras that cooperate by triangulation. The cameras are placed over the testing area, with a clear process overview. Placing reflective markers on the snake's joints gives the camera system the ability to track the snake robot in real time through space. Revealing a description of snake movement with down to sub-millimeter precision. Thus functioning as a reference to the less precise snake behavior described by tactile data. Figure 3.2 shows a simulation of how the cameras are configured. The figure is retrieved from a software platform called Motive, described in section 3.2.2. For a better understanding of the snake's behavior, and easier debugging, a visualization tool is developed. Importing all incoming data from both the snake and the camera system into a simulated environment leads to a more comprehensible display of data. Especially important is the data regarding collisions. It tells how the snake interacts with the environment. Collision data desirable to visualize include

- which joints are involved in collisions
- on which side of the snake the collisions occur
- tangent force data (e.g. magnitude and direction)
- normal force data (e.g. magnitude and direction)

Chapter 6 has this in mind when developing the sensor fusion module.

1.4 Problem Description

A snake robot (Mamba) has been developed [3]. Lateral propulsion by Perception-driven Obstacle-Aided Locomotion is being achieved and monitored by combining on-board sensory and an external camera system. Sensor fusion ensures a more reliable and robust robot behavior. This thesis focuses first and foremost on the visual perception and sensor fusion part of the project, but discussions regarding the tactile part of the project are included where it is deemed necessary.

Tasks

- equip joints with reflective markers and collect camera data for different marker configurations. Compare the deviances between actual and estimated joint positions, from kinematic calculations, based on the marker configurations

- spawn objects from the real world in a simulated environment (e.g. joints and obstacles), and visualize real events (e.g. collisions) and corresponding sensor data, in the simulated environment
- achieve a robust way of detecting collisions and gather the corresponding data. Use the camera system to find and visualize pushpoints
- integrate tactile data into the method of finding pushpoints (i.e. sensor fusion)
- achieve perception-driven obstacle-aided locomotion

1.5 Contributions

The bullet points below list the most important contributions to this project.

- implementing the visual perception module to collect and visualize all necessary data for perception-driven obstacle-aided Locomotion (i.e. collision points, collision forces direction, etc.)
- integration with the tactile perception module (i.e. sensor fusion.)
- performing real experiments with POAL (replicating the simulation results that were previously achieved in [1].)
- maintaining and documenting source code with the aim of releasing the framework as an open-source project

1.6 Task Restrictions

This thesis focuses on Obstacle Aided Locomotion (OAL) in the horizontal plane and thus assumes that all contact forces are applied to the sides of the snake. All testing takes place in an inside environment with clearly defined obstacles.

1.7 Report Outline

- **chapter 1** contains a brief introduction regarding the project. Background and motivation are described together with tasks and contributions. Task restrictions are also mentioned
- **chapter 2** contains previous work relating to this project. A discussion of how it relates is included at the end
- **chapter 3** describes the tools and methods used to develop the system
- **chapter 4** shows a high level and low-level overview of the system. Descriptions of system elements are included

- **chapter 5** describes the implementation and development of the system's visual perception module
- **chapter 6** describes sensor fusion and the approach for achieving POAL
- **chapter 7** is about the experimental setup and execution of the tests conducted
- **chapter 8** displays the results of conducted experiments
- **chapter 9** discusses the results of conducted experiments and several aspects of the project based on observations
- **chapter 10** contains future work
- **chapter 11** contains the project conclusion

Chapter 2

Literature Review

A series of papers related to the topic of this thesis is presented in this chapter. The papers are presented in sections 2.1.1 - 2.1.3, and the most attractive parts are discussed in section 2.1.4.

2.1 Related Work

2.1.1 Obstacle Handling

In a paper published [10], obstacle avoidance is explored with a snake robot moving through a narrow hallway. The goal is to traverse the hallway without touching any mid path hindrances or walls. An idea of creating a map of the environment by utilizing binocular stereopsis or laser ranging is presented in the paper. Creating maps of local surroundings is a useful feature when developing autonomous robots and is discussed in 2.1.4.

In a paper more related to this thesis' obstacle handling [11], a more relaxed approach to obstacle oriented locomotion is presented. Instead of a strict non-collision policy presented in [10], the system allows for contact between a robot and an object in a controlled manner. This opens up the possibility for fast navigation through cluttered terrain without damaging the robot.

Based on an investigation of lateral undulation in nature [12], at least three pushpoints must be found, on alternating sides, to acquire motion. A conceptual drawing of the criteria is found in figure 6.4. [13] reflects over the sufficiency of knowing the position of the collision points with a resolution limited to link size. The reason being the lack of believed significance of sub link force location compared to the location of force with respect to the entire snake. The legitimacy of the claim is validated in [14], as well as in this thesis, through experimental results from testing POAL with a downsized resolution.

2.1.2 External Data Gathering

In [8], researchers are aiming at making a robust robotic system to help rescue workers in urban disaster areas in the search for survivors. It mentions the handling of previous disaster events like the World Trade Center (WTC) disaster scene. In the aftershock of the WTC disaster, robots equipped with cameras were deployed. They were assigned to places restricted to human traffic due to fire hazards or structural instabilities. The problem with the deployed robots was their lack of 3D image capturing abilities. This made identification very difficult, often mistaking objects of concrete etc. for victims. The lack of 3D imaging also resulted in the robots getting stuck more than two times per minute.

The authors' approach is to make robust 3D sensors that can map unknown areas and locate victims efficiently by investigating the concept of structured light. The concept of structured light is based on an active light source that emits a known light pattern. A camera mounted at a known, fixed distance from the projector registers any deviance in the point cloud pattern emitted due to 3D structural inconsistencies. The paper shows through experimental results the feasibility of accurate information gathering by 3D mapping of indoor disaster environments. The robot used is a six-wheeled car and is dependent on having a relatively smooth surface to move.

The paper [15] concerns obstacle aided locomotion for snake robots. For robots to be able to operate autonomously they need to acquire information about their surroundings in a way that can be used to plan their actions accordingly. The paper lists three key challenges that need to be solved on the perception side of an autonomous robot in order to achieve obstacle aided locomotion

- *sensing*, the use of adequate sensory to capture needed information about the surroundings
- *mapping*, to organize the sensing output in order to create representations that can be exploited
- *localization*, to estimate the robot's pose in relation to the represented environment

Several important types of perception sensors are discussed, classified and put into respective branches of a tree structure based on their traits. The term POAL is adopted as locomotion where a sensory-perceptual system is utilized to gain knowledge about the snake's surrounding operational space, identifying walls and obstacles.

The authors of [16] have developed a six-legged, map generating robot called *LAURON V*. LAURON V uses a rotating laser scanner to make 3D panoramic point clouds of uniform angular resolution. Aiming to create an approach for mapping unexplored territory autonomously. The motivation behind the project is to able the robot to work even without real-time communication between the robot and an operator. Examples of use are in planetary exploration or search and rescue missions. Two environmental models are created. One model uses only the latest point clouds for fast local navigation. The other model contains a global map of the explored area by combining all available point cloud data. The global map is used for high-level path planning over longer distances. LAURON V updates its current position and orientation estimation based on the updated global map. LAURON V uses a visual type *odometry* approach to estimate its approximate location.

The authors of [17] implemented a structured light sensor on a robotic snake to enable it to do tasks autonomously. The paper focuses on analyzing the point clouds to locate specific environmental features, in this case, poles. Autonomous pole climbing behavior is achieved, demonstrating the possible autonomy enabled by improved sensing.

Communication is an important aspect of activities such as search and rescue. A fast and robust way of message transmission is crucial for such operations. A paper published [18] proposes the use of mesh networks for reliable and fast communication for instance in urban areas or in underground locations. A mesh network is a self-organized and self-configured form of network. Infrastructure nodes connect directly, dynamically and non-hierarchically to as many other nodes in the network as possible. Establishing an efficient way of routing data and maintains network connectivity even in cluttered areas.

2.1.3 Sensor Fusion

Learning about different types of terrain and footholds can be of interest for autonomous robots. The authors of [19] aim to use terrain templates to teach a robot foothold rankings when traversing rough terrain. The authors use a four-legged robot called *LittleDog* for development and testing. The robot learns where to step based on the foothold rankings.

[20] discusses the use of *dead-reckoning* based on on-board navigation sensors like Inertial Measurement Units (IMU). Dead-reckoning is not suited for long-term navigation because of drift caused by unavoidable accumulated errors. Updating position estimations by using external references for correction is a way to stop drift. Examples of external references are for instance GPS receivers or cameras. They are tools for knowing absolute position. A drawback with absolute position sensors is the low rate measurements for delicate navigation. Combining the two methods creates a high-accuracy system with a high updating rate. The paper [20] mainly discusses localization and navigation algorithms by utilizing encoder, compass, IMU and GPS measurements integrated with the Extended Kalman Filter (EKF).

2.1.4 Discussion

The list of challenges that needs to be solved in order to achieve POAL in [15] is a good base for technical development, as seen with the sensing robotic snake in [17]. A variety of options are available when choosing a sensory organ. This thesis focuses on external, passive stereo vision in combination with on-board force sensors. Future advancements consider ideas where both mapping and localization are taken into account, see section 10.2.

The concept of a robot mapping its surroundings in [16] and [10], and storing the information can be of use when utilizing on-board cameras or alternative, local perception methods. Section 6.3 explains this thesis' method for finding pushpoints without the necessity of physical contact between the snake and the obstacles. Section 10.2 describes a futuristic pushpoint localization scheme similar in nature to the one in section 6.3. Knowledge about the snake's local surroundings in reference to its estimated joint positions can work as a substitute for the external camera system implemented in this project. A joint does not need to be in contact with an obstacle for the snake to know it is a valid pushpoint. Virtually making a pushpoint extractor from vision without seeing the obstacles in real time. As mentioned, locating pushpoints with the help of on-board cameras is described more in depth in section 10.2.

The legged and wheeled robots in [8] and [16] are restricted to follow paths that are not too irregular. A camera mounted on a snake-like robot can traverse landscapes void of resemblance to a road or a smooth surface. That is an enormous advantage when searching through wreckage. A relatively homogeneous, streamlined body minimizes the chance of it getting stuck [13]. Robotic snakes can be developed with modularity in mind. Further reducing the chances of getting stuck. A possible design can be to create a system of independent links that can join together and split up depending on what the situation demands. Dividing into subsystems, each subsystem consisting of a few links, can make search and rescue missions substantially more efficient. Placing relays in their paths can ensure communication even underground. With a high enough number of entities in the field, it becomes profitable for mesh networks to be constructed. Mesh networks are robust networks ideal for communication in complex environments. Section 10.4 discusses this topic more in detail.

Teaching a robot the best paths to take based on previous experience, as in [19] can be of great use also when dealing with robotic snakes. Though a legged robot may benefit more from this because of its instability in the case of a misstep. Snakes are as a default in a very stable position due to their area of contact being large in comparison to leg based robots. The larger the contact surface the more the robot can ignore small imperfections in the terrain. Snakes may therefore work better in rough terrain with regards to stability.

On-board and external sensory in combination with EKF [20] eliminate the drawbacks of using one or the other type of sensory. This feature comes in handy when dealing with autonomous robots. An example of an area where this would be an improvement is found in section 10.4.

Chapter 3

Tools and Methods

Several computer programs contribute to the development of the finished product. An introduction to the different programs follows in this chapter together with a description of the snake robot used.

3.1 Robotic Snake Setup

The project utilizes a robotic snake called *Mamba* [3], see figure 5.7. Figure 5.7 is the model for the displayed snake setups for all simulations in this report, and figure 3.1 shows the joint numbering standard for all simulations.

The snake consists of 14 links interconnected through 13 joints. See table 3.1 for joint module specifications.

Table 3.1: Joint module specifications. Gathered from [3].

Parameter	**Value**
Weight	$310g$
Height	$70mm$
Width	$70mm$
Length between joint axes (link length)	$89mm$
Degrees of freedom (DoF)	1
Maximum joint travel	$\pm 90°$
Maximum continuous joint torque	$2.3Nm$
Maximum joint speed (without load)	$429°/sec$

Seen in table 3.1 each joint is able to rotate $\pm 90°$ about one of its axes. This implies planar snake attitude when all joints are oriented the way as seen in figure 5.7. In previous projects, every other link has been rotated 90° about its own x-axis compared to the adjacent ones. Resulting in a snake robot that is able to move relatively freely in 3D space. Because of the project restriction of strictly 2D movement, all links have the same orientation about their x-axis and y-axis, and are only capable of rotation about the global z-axis. Each joint is equipped with strain gauges to detect contact between the snake and its surroundings. Strain gauges are tuned by adjusting 18, integrated *potentiometers*, see [14] for clarity about what the potentiometers do. Three red cylinders, seen in figure 5.7, work as clearly defined obstacles. The cylinders are mounted to a board representing the ground. They are of similar build, with a 10 cm radius and a height of 20 cm.

Illustrations of the kinematic chain throughout this report depict joints as rectangles, like in figure 3.1. Not to be mistaken for links. The reason for depicting joints in this fashion being that all the snake's sensory is located in the joints. The snake's links are the parts stretching between the joint centroids. Note that this is not the case for figures of the snake spawned in Gazebo or the snake exported from Gazebo to RViz where joints are located in between the cubes representing links, see figure 3.7 and 3.8. The reason for the different depictions has to do with the different types of sensory used for the real and simulated snake. This is detailed in section 9.4.

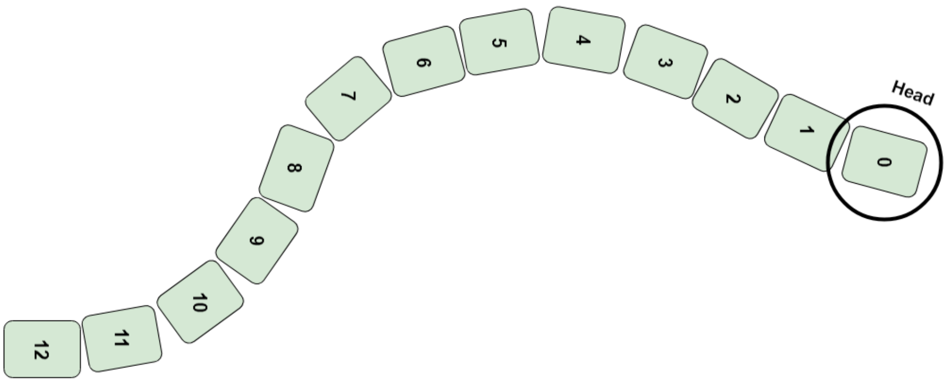


Figure 3.1: The joint number representation standard for all simulations in this thesis. Joint 0 represents the snake head and joint 12 serves as the tail.

To track the snake robot in a global frame, and to create a more reliable system, it is believed necessary to utilize cameras as external observers. Section 3.2 describes the workings of the camera system used.

3.2 Camera System

Visual perception is introduced through a setup of 16 collaborating cameras from *OptiTrack* overhanging the testing area. See figure 3.2 for the simulated representation of the camera configuration as seen in the associating OptiTrack software, *Motive*. The moti-

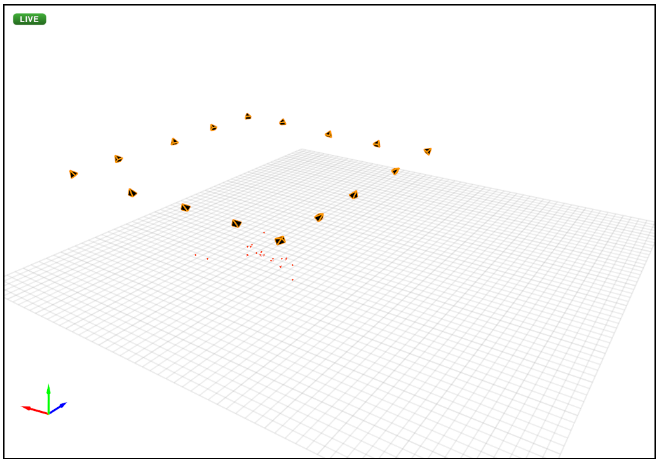


Figure 3.2: 16 cameras overhanging several markers (red dots) as seen in Motive, described in section 3.2.2.

vation behind the chosen camera system is the easy to use integrated software elements from the same developer, *NaturalPoint*. The elements are presented in the following subsections.

3.2.1 NaturalPoint/OptiTrack

The camera model used for surveillance is a product from *NaturalPoint* [21] called *OptiTrack Flex 13* [5], see figure 3.3. The camera produces an image size of 6.1444×4.9152 mm, and has an image resolution of 1280×1024 (1.3 Megapixels). It has a frame rate ranging from $30 - 120$ FPS and a latency of 8.3 ms. Input power and data transfer are done over the USB 2.0 protocol.

Figure 3.3: The OptiTrack camera, Flex 13, used for tracking purposes [5].

Combining the Flex 13 cameras with OptiTrack's proprietary real-time processing algorithms make them able to track motions within a tolerance of less than 0.5 mm. This makes OptiTrack ideal for rigorous tracking applications such as complex motion tracking. OptiTrack cameras are used in combination with associated software for handling camera data. It is called *Motive* and is described in the next section.

3.2.2 Motive

Motive is a software platform designed to control motion captures for a line of tracking applications and purposes. It uses reconstruction, the technique of compiling multiple 2D images of reflective markers to obtain 3D coordinates [22]. The reflective markers mentioned are spheres with a diameter of approximately 1 cm, coated with reflective tape, see figure 3.4. The cameras detect the infrared radiation that bounces of the reflective markers when light hits them. In Motive, a registered marker is represented as a red dot in the estimated location in space, see figure 3.2. It is not possible to register rotations of single markers, and they have no ID attached to them for identification. Ergo, using marker streaming data from produced from camera tracking is a difficult task. Grouping clusters of three or more markers together creates a *rigid body* (RB), assuming that the spatial relationship among the attached markers remains unchanged. Rigid bodies extend outward from a single point, the centroid, shown in Motive as the Rigid bodies' middle spheres. Motive is able to know the *pose* (position and orientation) of each of the RBs. RBs are also tagged with an ID and a name, resulting in a clear understanding of which RB is which.

Figure 3.4: Reflective marker used in tracking.

RB markers are actually expected marker positions from estimations in Motive. They appear as transparent spheres within the rigid body and represent the expected position of a reconstructed marker, see figure 3.5. From figure 3.5, the red areas are where the actual marker data experiences a slight deviation from its expected position. Red markers in the RB, *joint_12*, are examples of this happening. Deviations at this scale are containable, but further drift will cause markers to be unlabeled due to the mismatch between marker data and expected marker position. This leads to the loss of data.

Motive gives the ability to collect marker information. The next step is to use this information. Using the rigid body IDs, position and orientation data are streamed from Motive to the *node mocap_optitrack* in the application for robot control, Robot Operating System (ROS). See figure 4.2 for an overview of the data stream. ROS handles

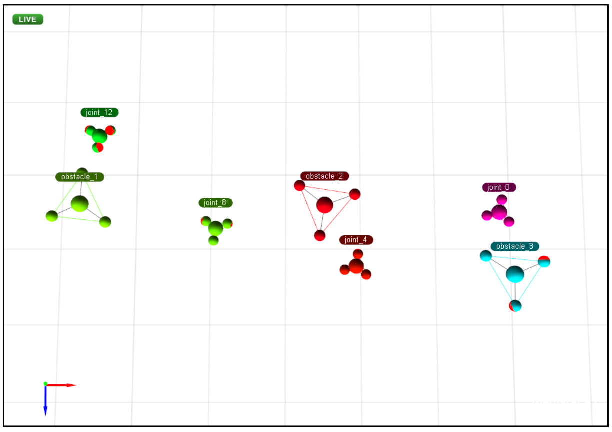


Figure 3.5: A picture of constructed rigid bodies. Three markers are joined together, creating a fourth sphere in the middle, representing the centroid of the rigid body. Any deviation between actual and estimated marker position data is represented in red, see joint_12 and obstacle_3.

incoming data from multiple programs through these nodes. It is explained in the next section.

3.3 Robot Operating System (ROS)

Robot Operating System is an open source collection of software frameworks that provides libraries and tools for managing code across multiple computers. It also has logic for message passing between processes, package management and more to help software developers create robot applications. This makes ROS a great candidate for this project's system development. Other platforms like *Mobile Robot Programming Toolkit* (MRPT) and *Microsoft Robotics Developer Studio* (MRDS) are alternatives to ROS depending on the use, but ROS is used here for instance because of its abundance of packages, its fast-growing community and as mentioned, ROS being open source. The developers are themselves saying that ROS is built from the ground up to encourage collaborative robotics software development [23]. Making ROS ideal for open source projects.

The main supported programming languages are C++, Python and Lisp, but at an experimental level, it supports both Java and Lua. ROS runs on Unix-based platforms and it is primarily tested on Ubuntu and MAC OS X. Different distributions of ROS are on the market. In this project, ROS Indigo is used.

ROS comprises of *nodes* that are in charge of controlling one or a few, specific tasks. A node can for example be in charge of path planning, or performing localization. The nodes are connected to each other through communication lines called *topics*. Topics are intended for unidirectional streaming communication. Nodes that generate data *publish* the data to a relevant topic. Nodes are not aware of who they are communicating with.

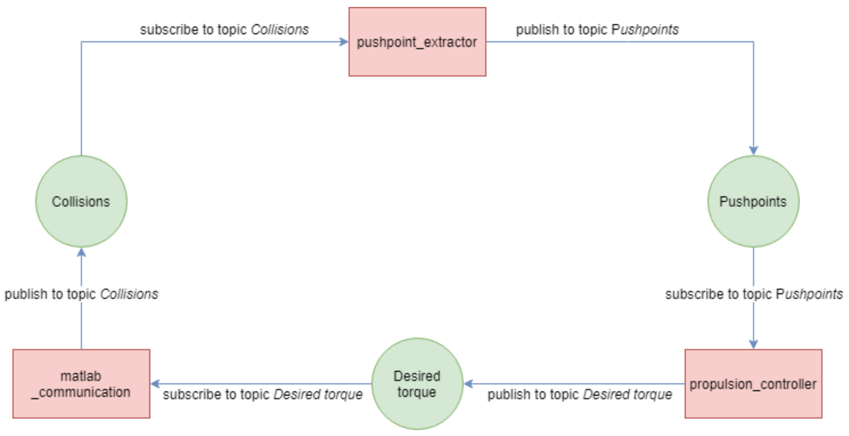

subscribe to topic *Collisions*
pushpoint_extractor
publish to topic *Pushpoints*
Collisions
Pushpoints
publish to topic *Collisions*
subscribe to topic *Pushpoints*
matlab
_communication
subscribe to topic *Desired torque*
Desired
torque
publish to topic *Desired torque*
propulsion_controller

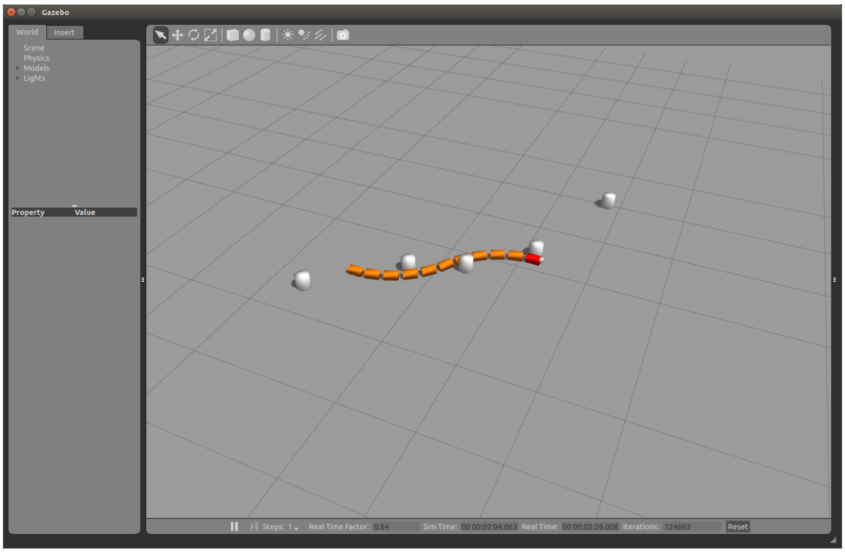
Figure 3.6: A representation of how nodes communicate with each other. Nodes are red rectangles and topics are green circles. Node *pushpoint_extractor* subscribes to the topic *Collisions*, which contains data published by node *matlab_communication*. Node pushpoint_extractor publishes data on the topic *Pushpoints* which is subscribed to by node *propulsion_controller*. There is no chronological sequence for publishing and subscribing. Node pushpoint_extractor can publish to the topic Pushpoints simultaneously with node matlab_communication publishing to topic Collisions.

Instead, nodes that are interested in data *subscribe* to the topics containing the desired data, see figure 3.6 for an example of the node/topic relationship. The figure shows an excerpt from the ROS overview in chapter 4. There can be constructed multiple subscribers and publishers to a topic, as seen in the more complex node/topic relationship illustration in figure 4.2.

ROS works as a platform for gathering, processing and distributing data. A clever use of incoming data is to display it in a simulated environment like Gazebo or RViz. Real data from ROS is therefore sent to both of the mentioned.

3.4 Gazebo

Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate robots in complex environments [24]. In addition to simulations, it also offers a physics engine with a much higher degree of fidelity when it comes to physics simulations than regular game engines. Gazebo is thus a respectable choice for snake simulations that include real-world data. The snake's specifications and the environment it is traversing can be described at a very precise level. Figure 3.7 shows the platform including a snake simulation.



Steps: 1 Real Time Factor: 0.84 Sim Time: 00 00:02:04.663 Real Time: 00 00:02:26.008 Iterations: 124663

Figure 3.7: Gazebo window with simulated snake and obstacles. Red link marks the head. The white object attached to the snake's head is a camera module not used in this project.

3.5 RViz

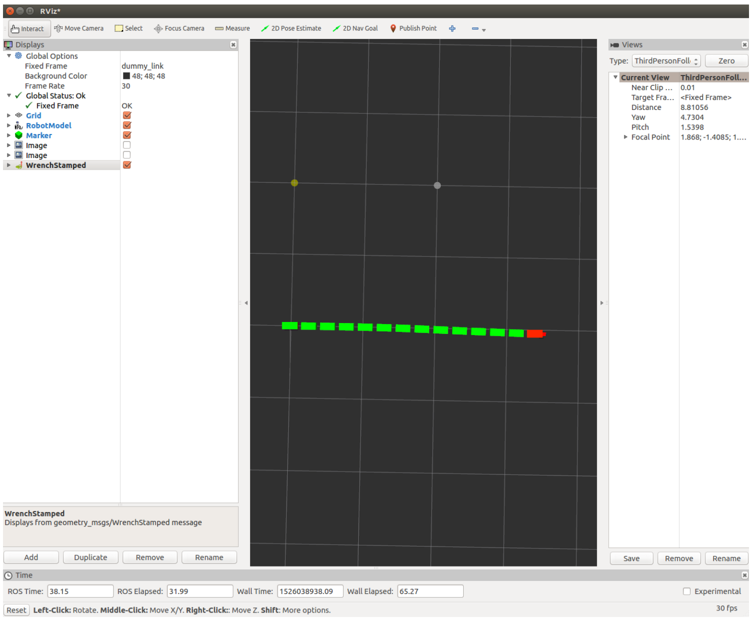
RViz is an open source 3D visualization environment that lets the user know what a robot sees, feels and does. This kind of insight makes RViz a great tool for tracking system behavior and to debug robot applications. The user can decide what to display and what information to hide by checking off boxes in the left screen panel, see figure 3.8. Previously RViz was used for showing simulated snake data from Gazebo [1]. RViz is here used to visualize the real snake setup and necessary information regarding collisions. Figure 3.8 shows the RViz environment together with a spawned snake from Gazebo.

3.6 LabVIEW and MATLAB

LabVIEW is a designer platform and development environment created by National Instruments. It is suited for tasks that require communication between hardware and software. LabVIEW is relevant when introducing tactile data, from the snake, in the simulations. Tactile data must be gathered through LabVIEW for further use, like sensor fusion.

The already existing LabVIEW interface is used here to be able to keep the work that is already done prior to this thesis. The snake controller in LabVIEW works as a node and is easily replaceable in the future.

MATLAB is an *IDE* used here for structuring and altering data coming from LabVIEW,



Global Options	
Fixed Frame	dummy_link
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
RobotModel	✓
Marker	✓
Image	☐
Image	☐
WrenchStamped	✓

WrenchStamped
Displays from geometry_msgs/WrenchStamped message

Type: ThirdPersonFoll Zero

Current View	ThirdPersonFoll...
Near Clip ...	0.01
Target Fra...	<Fixed Frame>
Distance	8.81056
Yaw	4.7304
Pitch	1.5398
Focal Point	1.868; -1.4085; 1....

ROS Time: 38.15 ROS Elapsed: 31.99 Wall Time: 1526038938.09 Wall Elapsed: 65.27

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Move Z. **Shift**: More options.

Figure 3.8: RViz window with a visualized snake imported from Gazebo. The red link marks the head. The smaller red object attached to the red link is a camera module not used in this project.

before passing it over to ROS. It also functions as a port for control input from ROS. Several plots have been made using MATLAB.

Chapter 4

System Overview

An overview of the workings of the system is presented in this chapter, highlighting the most important contributions and utilized parts. A high-level description of all system software components is presented in section 4.1, and a lower level presentation of the program is included in section 4.2, showing important, internal ROS interactions between nodes.

4.1 High Level Program Overview

Figure 4.1 depicts interactions between the programs integrated in this project. Arrows moving towards and from the rectangle representing the ROS module are subscribed and published information over topics, respectively. Yellow squares represent the sensing part of the system. All information is gathered here. Blue blocks are programs used for data handling and green circles show the nodes between ROS and connected programs. Blue blocks with thicker, black frames are the ones focused on the most in this thesis. The remaining blocks without thicker frames are covered in [14].

4.2 ROS Node Interactions

A map over the most relevant ROS nodes and topics is presented in figure 4.2. There are other nodes implemented as well, but they do not play an appreciable part and are discarded from the figure. Nodes with thicker edges are the ones focused on in this thesis. The remaining nodes are covered in [14].

The node *mocap_optitrack* is in charge of receiving the streamed rigid bodies from Motive. *visual data_topic_collector* takes in obstacle and joint poses from *mocap_optitrack*,

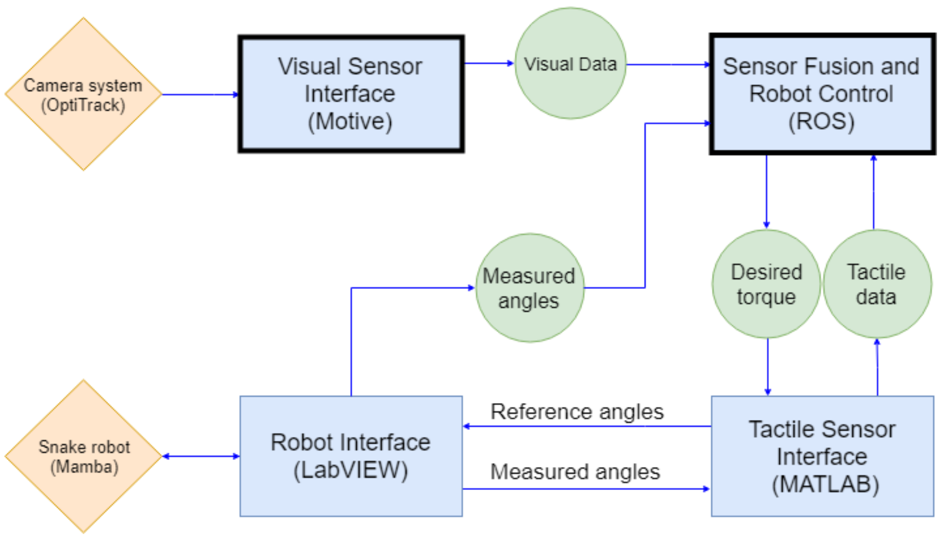

Camera system (OptiTrack)
Visual Sensor Interface (Motive)
Visual Data
Sensor Fusion and Robot Control (ROS)
Measured angles
Desired torque
Tactile data
Snake robot (Mamba)
Robot Interface (LabVIEW)
Reference angles
Measured angles
Tactile Sensor Interface (MATLAB)

Figure 4.1: High level overview of the different programs that are used in the project and their interactions. Figure 4.2 expands the ROS module, revealing the internal nodes and topics.

sorts them into two separate lists and publishes them on individual topics (i.e. *Obstacle positions* and *Joint poses*). *Joint poses* contains four *checkpoint joints*. Checkpoint joints are joints from Motive that function as reference joints when calculating the snake's kinematics, see section 5.1.1. *kinematics* subscribes to the topic *Joint poses*, processes the data as seen in section 5.1.1, and publishes the complete snake kinematics on the topic *Snake pose*. *Snake pose* now contains the estimated position of every joint. *matlab_communication* subscribes to the topic in question together with *Obstacle positions*, *Tactile data* and *Desired Torque* (the last one is a focus in [14]), outputting information about eventual collisions. Pushpoints are generated based on this outputted information. Collision data published on *Collisions* is subscribed to by *pushpoint_extractor* which publishes the extracted pushpoints on the topic *Pushpoints*. A more specific description of how pushpoints are found is seen in chapter 6. *visualizer* subscribes to the topics *Snake pose*, *Obstacle positions*, *Tactile data* and *Pushpoints* for visualizing the snake, obstacles, forces and pushpoints respectively.

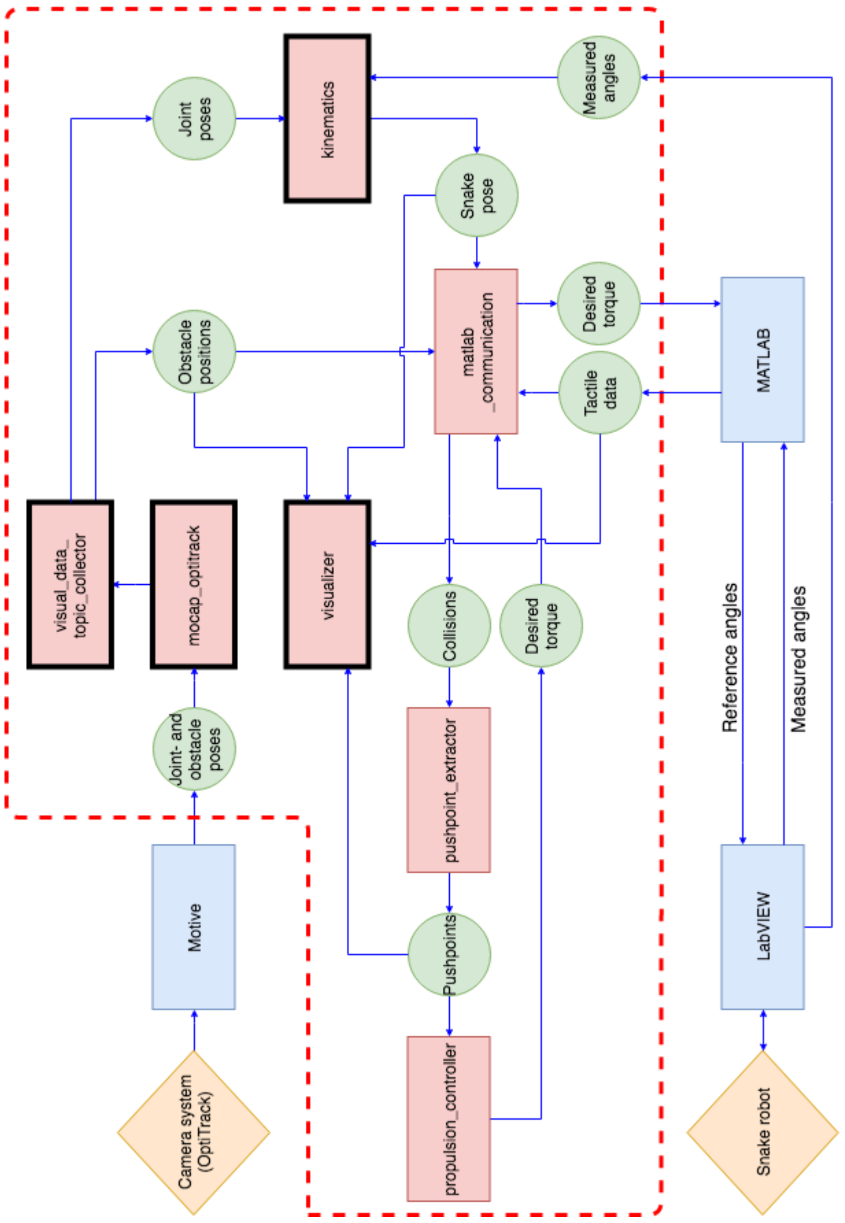


Figure 4.2: Inspection of the most important ROS nodes, shown as red rectangles, and corresponding topics, shown as green circles. Nodes highlighted by thicker edges are the ones most focused on in this thesis. Yellow squares represent the sensing part of the system and blue rectangles are programs used for data handling.

Chapter 5

Visual Perception

A thorough description of the implementation of the visualization module is presented in this chapter. Figure 5.7 is the used setup for all simulations depicted and figure 3.1 is the followed joint numbering standard.

5.1 Spawning Snake Setup in RViz

The purpose of implementing the camera module is to use simulations for pushpoint verification. This section contains the approach for simulating the snake and display necessary data for understanding what the snake is experiencing.

5.1.1 Kinematic Model

The mathematical approach for deriving the snake kinematics is given in this section. The kinematic calculations are influenced by the results of experimenting with marker configurations, see section 8.1. Different marker configurations are tested to see which one gives the least deviance between calculated and actual joint positions due to propagation error. Experimental results conclude with the use of four checkpoint joints, see figure 7.1c and 7.1b. As stated earlier, checkpoint joints are joints with known poses, functioning as reference points. They define the basis from which the kinematic calculations spring out of.

Since the obstacles used are standing cylinders, their orientation is trivial information. They each contain a marker triplet, placed on top of them, and are spawned in RViz directly from position data collected by OptiTrack, processed by Motive.

Both position and orientation are needed to correctly describe the snake joints. A transformation matrix is used to calculate the position of each joint relative to one another. The transformation matrix takes in the joints' measured angles and the link length (the distance between two interconnected joints). Calculating the kinematics from joint *m* to joint *n* is performed by the generalized transformation matrix [25]

$$H_n^m = \begin{bmatrix} \cos(\sum_{i=m}^{n-1} \psi_i) & -\sin(\sum_{i=m}^{n-1} \psi_i) & 0 & -l(\sum_{j=0}^{n-1} \cos(\sum_{i=m}^{n-j} \psi_i)) \\ \sin(\sum_{i=m}^{n-1} \psi_i) & \cos(\sum_{i=m}^{n-1} \psi_i) & 0 & -l(\sum_{j=0}^{n-1} \sin(\sum_{i=m}^{n-j} \psi_i)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (5.1)$$

where ψ_i is the measured joint angle for joint i and l is the link length.

Joints 12, 8, 4 and 0 have poses coming directly from Motive and are consequentially seen as checkpoint joints. Marked as red in figure 5.1. Transformations from checkpoint joints to their adjacent joints are done, finding the adjacent joints' positions relative to the checkpoint joints. Adjacent joints are colored green in figure 5.1. Another transformation is executed from checkpoint joints to *middle joints*. Middle joints are the joints right between two neighboring checkpoint joints and correspond here to the white joints 2, 6 and 10 in figure 5.1. Middle joints are calculated from both sides. The resulting middle joint position value is the mean of two transformations from opposite sides. The position of joint 2 is for instance the mean of the transformations from joint 0 and 4. This way of calculating middle joints reduces the propagation error even more by affirming their positions from multiple stands.

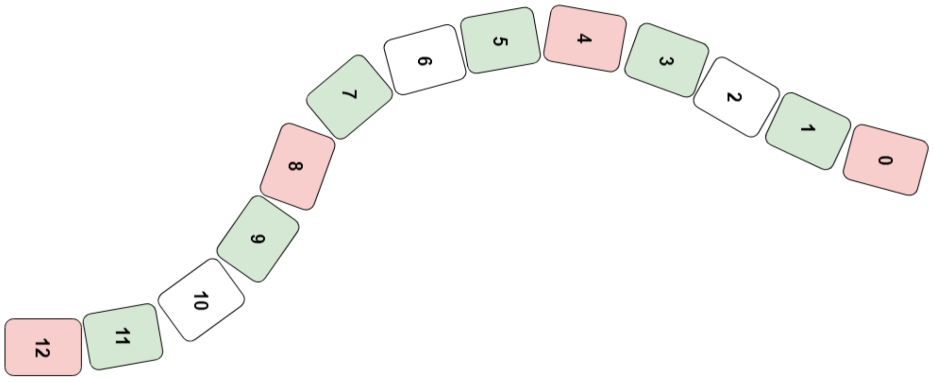


Figure 5.1: Snake model showing classes of joints used for calculating the snake's kinematic chain. Red joints are checkpoint joints, green joints mark all adjacent joints to a checkpoint joint and white joints represent middle joints (joints right between two checkpoint joints).

Simulation after Kinematic Calculation

Real data such as measured joint angles is not yet included in the calculations done in section 5.1.1. Therefore, the kinematic calculations which are based on using measured joint angles, are consequentially incorrect. Only the reference joints from Motive are accurate in both orientation and joint position. Figure 5.2 shows the four checkpoint joints, one every four joint, together with simulated obstacles. The simulated joints and obstacles have genuine poses compared to reality. In the figure, the snake follows the snake convention from figure 5.7, and moves from left to right.

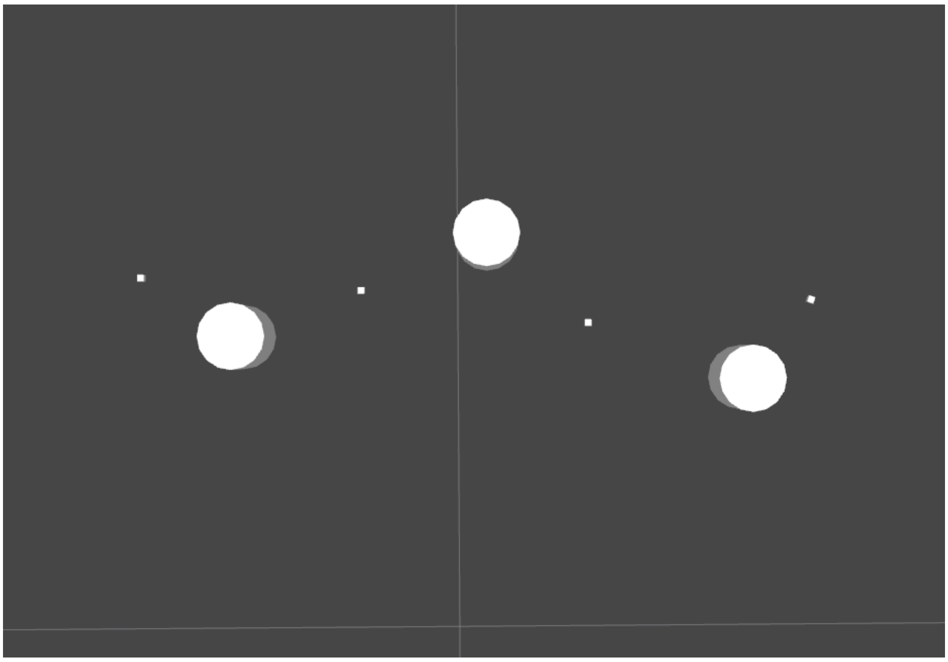


Figure 5.2: Resulting snake simulation after applying calculations for position estimation. Measured joint angles are not included. For this reason the estimated joint positions are incorrect and only reference joints are showing. These joints come straight from Motive and are accurate in both orientation and position.

5.2 Including Real Sensor Data in RViz

Measured joint angle data and forces acting on the snake are integrated with RViz to get a more complete understanding of the snake's behavior. The integration procedures are described in the following subsections.

5.2.1 Including Joint Angle Data

Equation 5.2 combines the quaternion representation of the Euler rotations [26]. A conversion from Euler angles to quaternions ensures the handling of singularity cases also known as gimbal lock. Gimbal lock is a specific orientation that results in the loss of one degree of freedom in a three-dimensional system. It is not directly applicable to this thesis but is considered for future project development.

$$\mathbf{q} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix}. \quad (5.2)$$

Since the snake operates in a 2D environment, the only axis of rotation is the z-axis. Therefore the only angle with the ability to step away from a zero value is the *yaw* angle, ψ. Both *roll* and *pitch*, ϕ and θ respectively, keep their zero values as no rotations about the x- or y-axis are possible. This restriction in movement reduces the matrix in equation 5.2 to

$$\mathbf{q} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} \cos(\psi/2) \\ 0 \\ 0 \\ \sin(\psi/2) \end{bmatrix}. \quad (5.3)$$

The new values found are included in RViz, synchronizing the simulated and the real joint angles. ψ is sent as a list of measured joint angles from LabVIEW and is subscribed to by the node *rvizpublisher*. Yaw is therefore set directly from registered snake values in real time. Figure 5.3 displays the end product of the rotations. Green squares are snake joints and white cylinders are obstacles. A look at figure 5.3 reveals authentic joint orientations. Real joint orientations result in a correct calculation of the kinematics which again gives correct joint positions, as was predicted in section 5.1.1.

Only the joints connecting the links are seen in RViz, as mentioned in section 3.1. The tiny cube design makes testing and debugging more comprehensible because of its spacious style. A transformation of the joints to resemble the snake in figure 3.8 is a trivial procedure of changing cube dimension specifications in ROS. Because of what is stated in section 3.1 about the simulated snake in RViz and Gazebo being different in design compared to the real snake, transforming the joints to resemble the snake in RViz is undesirable. Notice that obstacles are smaller than in the real setup. This is also due to clearness and is easy to alter.

The series of figures 5.4a - 5.4c illustrates how well the snake simulated in RViz actually follows the real snake. Moving any of the snake's joints or any obstacle results in a similar motion in RViz, in real time.

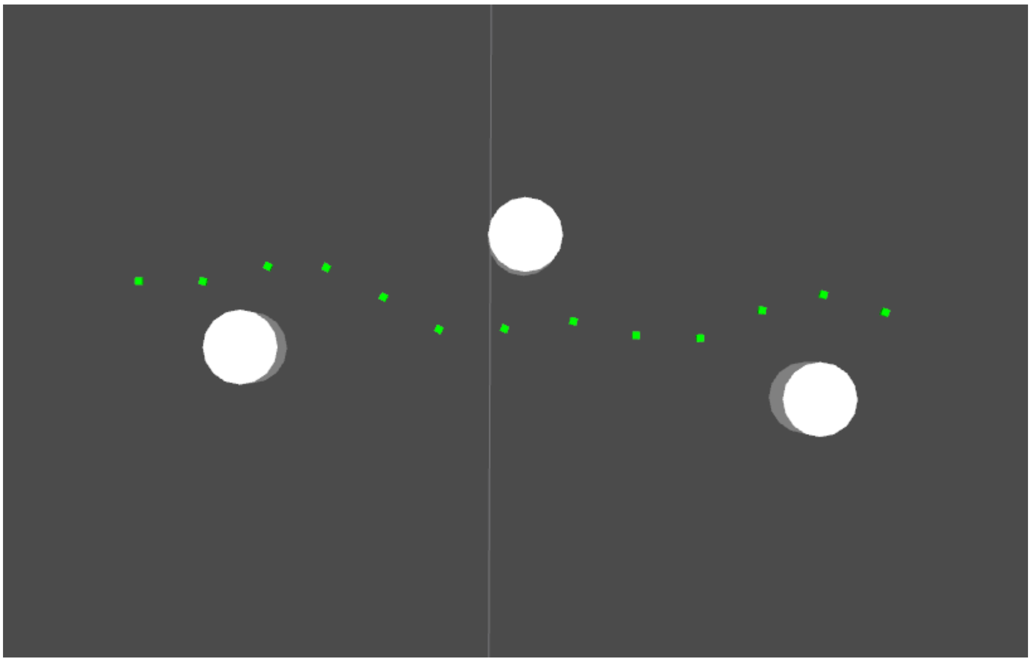
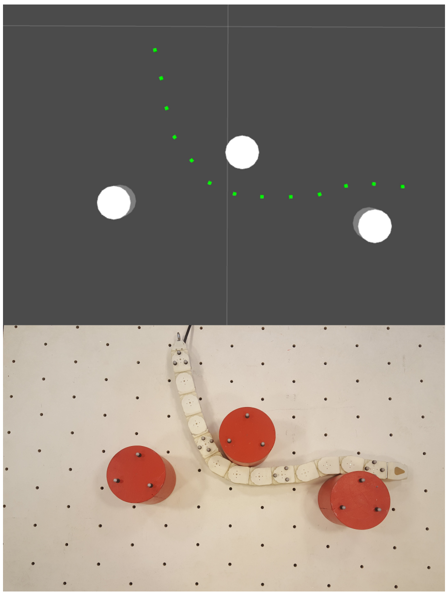


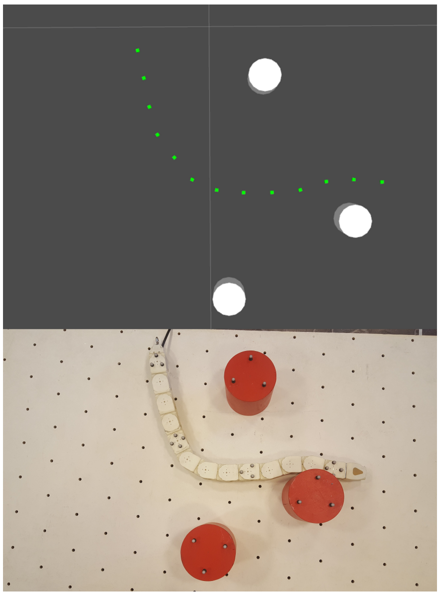
Figure 5.3: Joints point in the directions of real measured joint angles from LabVIEW. Consequentially, estimated joint poses are accurately displayed.

Figure 5.4a, 5.4b and 5.4c conclude with a degree of successful kinematic transformations in section 5.1.1, and system reliability is maintained. Note that RViz in figure 5.4c is not quite able to represent the snake with total accuracy. This has to do with the degree of pre-calibration of measured joint angles from LabVIEW, and resets of the Motive joints' heading. See section section 9.5 for a discussion on this topic.

5.2.2 Low Pass Filter for Unstable Rigid Bodies

The deviation between actual and estimated marker position data, as mentioned in section 3.2.2 can cause rigid bodies to vibrate between states, experiencing rapid changes in orientation. The physical upper limit to the rate of change in heading is $429°/sec$ as seen in table 3.1. This physical limit is not nearly as high as the rate of change in angles experienced in Motive as a result of the vibrations. Hence a logical restriction based on the joints' physical limits prevents the rapid changes in Motive to manifest themselves in the program. In practice, this is done by implementing a basic low pass filter. The constructed low pass filter works by denying the rate of change in heading to exceed a set limit. Hence improving the capabilities of the system by virtually steadying the unstable rigid bodies. In a perfectly working tracking scenario, such a filter would be superfluous. This is not the case here due to elements of light pollution etc., detailed in section 9.3.


(a) Initial snake at rest. Simulated snake showing similarity with real one.


(b) Obstacles are moved, showing the same motion in RViz.

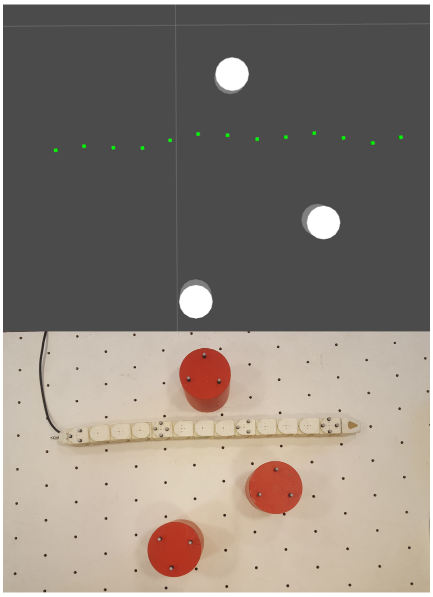

(c) Real snake is moved, resulting in a resembling movement in RViz.

Figure 5.4: A series of pictures showing movements of the real snake and RViz's attempt to imitate the movements.

5.2.3 Visualizing Actual Force Data

Actual forces are introduced in RViz to assist in an easier interpretation of what the snake experiences at a given time. Strain gauge data is imported from LabVIEW via MATLAB as seen in figure 4.1. The ROS node overview in figure 4.2 shows the *visualizer* node subscribing to the topic *Tactile data*.

Force direction is given by joint orientation, having tangent forces displayed rooted in the joint, running along the orientation of the joints. Normal forces center in the same place as tangents, pointing outwards from an obstacle in a perpendicular manner relative to the tangent forces, see figure 5.5. Based on the discussion of how precise the force picture's resolution needs to be in section 2.1.1, it is deemed sufficient to display acting forces on joint level.

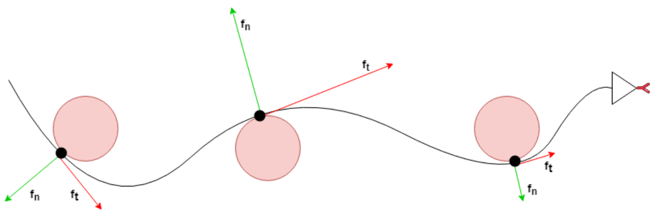


Figure 5.5: Conceptual model of collisions between a snake and obstacles. f_t and f_n are tangent and normal forces respectively. The larger the force the longer the arrows become

Strain gauge data ranges from $[0, 1023]$ [14], with 0 being force sensor saturation for forces experienced from the right side, and 1023 being the saturation from the left. Force magnitude is displayed as arrow length, figure 5.6. Arrow length is proportional to registered strain gauge data. On initialization all the joints' *zero force points* are imported into ROS [14]. Zero force point is the value of which no force is acting upon the joint. The zero force point varies depending on the tuning of the sensors' potentiometers. For perfectly tuned potentiometers, 512 is the zero force point. Arrow length is set to 0 for this value. New force data is subtracted from the zero force point and the resulting sign from the calculation decides normal force arrow direction. The attached video named *ForceTest.mp4* shows the simulated snake in RViz reacting to forces acting upon the joints of the real snake. Force arrows become longer as the push against the joints increases, as expected. The video is discussed in section 9.1.

Figure 5.6 differs from figure 5.5 in that not only the joints colliding with obstacles are experiencing forces, as seen in figure 5.5. The reason for this is the sensor type used and how these sensors are interconnected. It is discussed in section 9.4.

The joints of different color in figure 5.6 are presented in chapter 6 and are ignored for now.

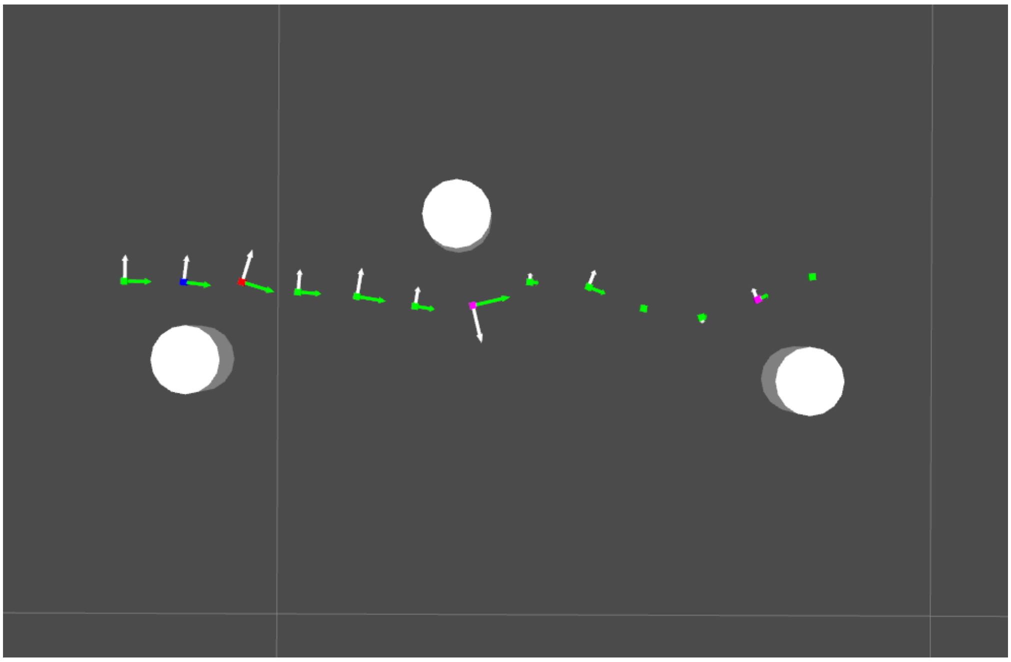


Figure 5.6: Strain gauge data is included in RViz. Force magnitude and direction for both tangent and normal forces are displayed as green and white arrows respectively.

5.3 Spawning the Snake Setup in Gazebo

This section describes the procedure of exporting the initial, real snake setup in Gazebo. The implementation is created as a groundwork for future development. Together with RViz, Gazebo can grant easy to detect indications of system behavior in real time.

A *.world file is used when initializing Gazebo and is loaded on Gazebo startup. The *.world file, created by [1], specifies the simulated world's physical descriptions regarding obstacle positions, initial position of the last snake joint and initial joint angles. Previous work specifies the joint's starting position to be in the center of the world and initial joint angles to have the default values stated in equation 5.4.

$$\psi_{init} = \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & -0.2 & -0.2 & -0.2 & -0.2 & -0.2 & -0.2 \end{bmatrix}^T \tag{5.4}$$

In order to spawn the real-life snake and obstacle poses in Gazebo, the *.world file is altered to contain obstacle and marked joint data from Motive, and measured joint angle data from LabVIEW. *snakebot_world_setup* is a node purposed for this task. It alters the details of the tail joint's starting position to be similar to the one captured by OptiTrack, together with the alteration of initial obstacle positions. The initial joint angles seen in equation 5.4 are changed to match the measured joint angles coming from LabVIEW. Launching *snakebot_world_setup* gives figure 5.8, the simulated version of the real snake in figure 5.7. Note that only the position of the end joint is considered in the *.world file. The remaining joints are introduced by estimating their positions based on link length and measured joint angles. For this to be precise, measured joint angles are required to be correct.

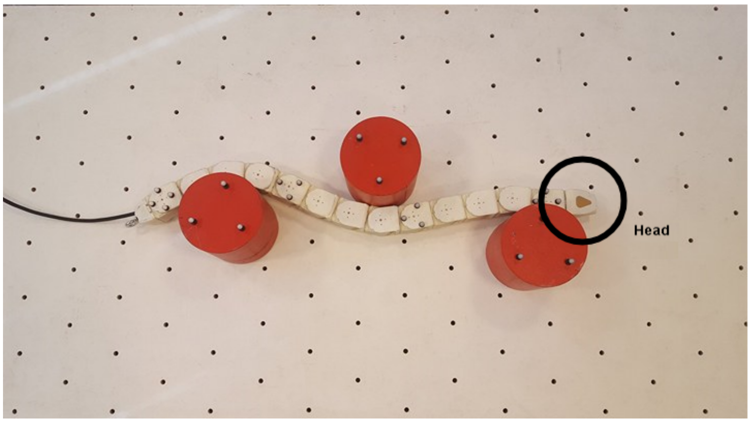


Figure 5.7: Real snake (Mamba) setup with obstacles . Spawning it in Gazebo yields figure 5.8.

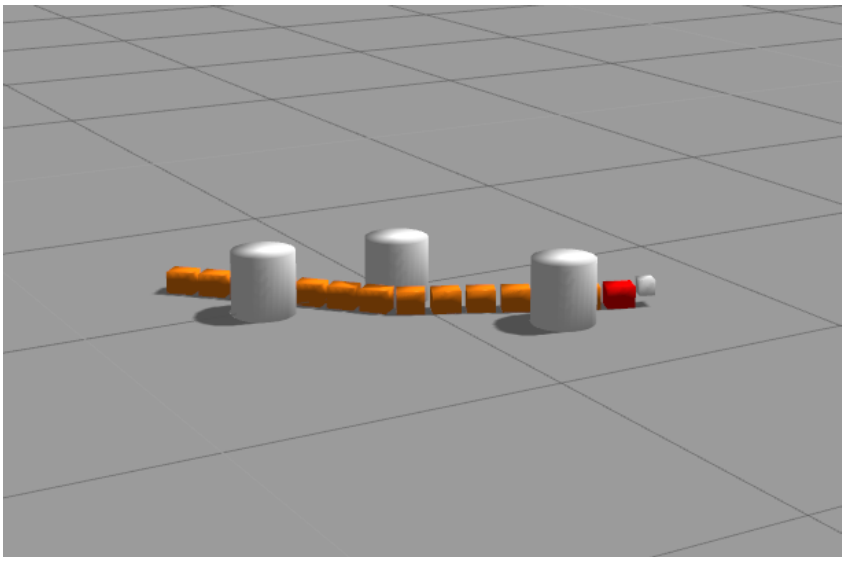


Figure 5.8: Simulated snake in Gazebo based on real setup in figure 5.7. Red link represents the head.

Chapter 6

Sensor Fusion for Perception-driven Obstacle-Aided Locomotion

This chapter explains the method of sensor fusion and the logic behind extracting push-points for POAL for this project by the use of sensor fusion. It is based on the developed visual model in chapter 5.

6.1 Sensor Fusion

Sensor fusion is the term for combining input from multiple sensors to synthesize a unified representation of the environment. That is, data verification from different perspectives. Sensor fusion is an umbrella term and can be involved in a system to different degrees. Following this section are a couple of basic examples of sensor fusion systems.

6.1.1 Biological Sensor Fusion

An example of sensor fusion is how the human sensory system, in combination with the brain, work together to perceive their surroundings. The human brain is analogous to a computer, processing data from multiple sensors like the ears, nose and eyes. Illustrated in figure 6.1.

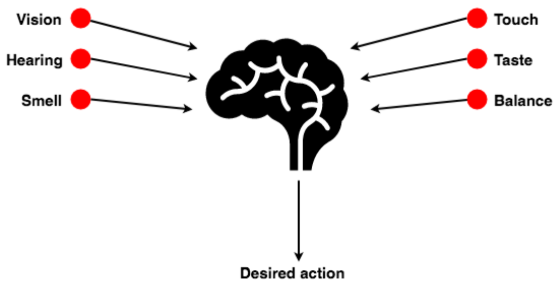


Figure 6.1: The human body is a system capable of completing complex tasks with the use of sensor fusion between multiple sensing devices.

6.1.2 Sensor Fusion Implemented in This Project

The degree of sensor fusion reliance in this project is limited. Instead of the numerous sensors listed for the human brain to process data from in section 6.1.1, only two sensor elements are used to gather data here. These are tactile data and visual data. This project aims to implement a basic form of sensor fusion where tactile data plays a role in verifying the correct found pushpoints. Together with the visual module, they perform a check to verify that the correct pushpoints are being found for achieving POAL. Visual data from the camera system (i.e. joint and obstacle poses) is compared to tactile data coming from the snake. Verified pushpoints by sensor fusion are obtained as a result. The basic workings of the implemented sensor fusion are seen in figure 6.2. A more detailed depiction is seen as the flowchart in figure 6.3.

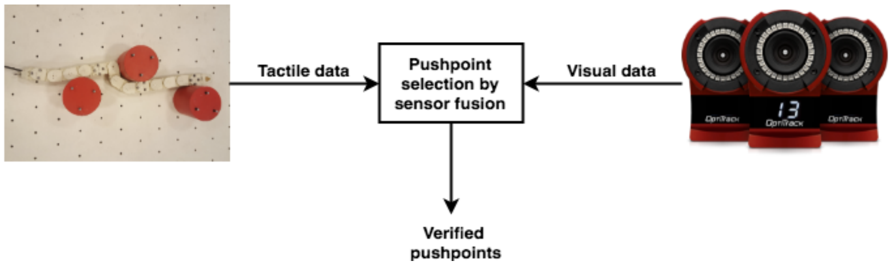


Figure 6.2: Tactile and visual data is combined to verify correct pushpoints from different perspectives.

6.2 Pushpoint Selection by Sensor Fusion

Figure 6.3 is a flowchart that shows the basic system strategy for finding pushpoints. It is investigated in more detail and referred to throughout this section.

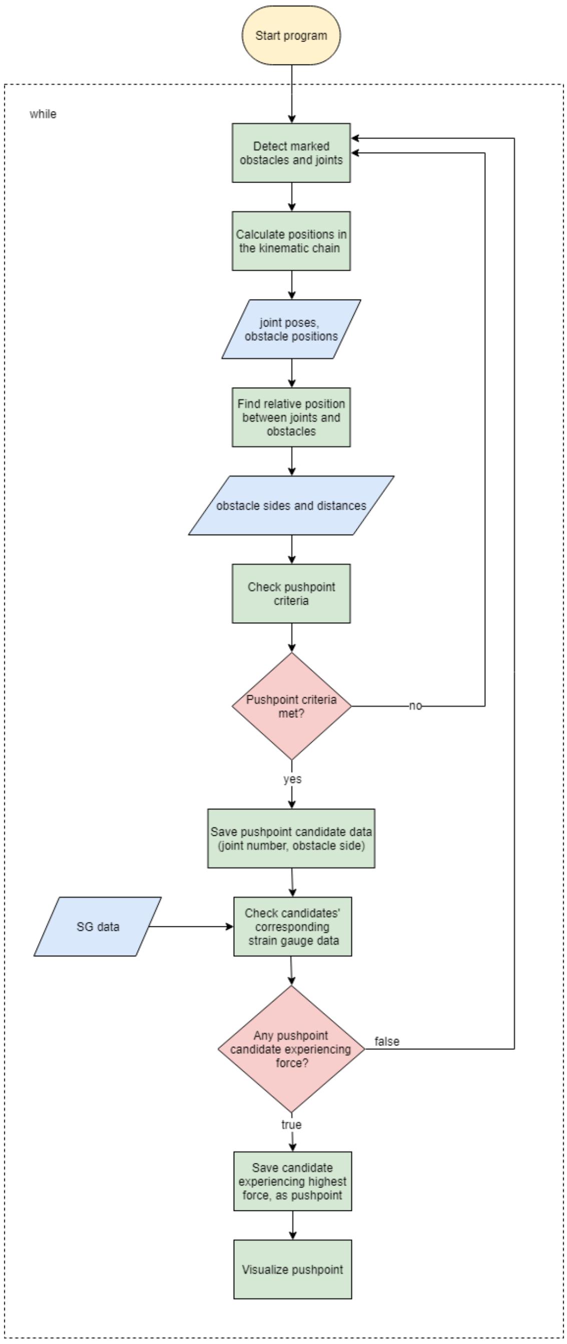

Start program
while
Detect marked obstacles and joints
Calculate positions in the kinematic chain
joint poses, obstacle positions
Find relative position between joints and obstacles
obstacle sides and distances
Check pushpoint criteria
Pushpoint criteria met?
no
yes
Save pushpoint candidate data (joint number, obstacle side)
SG data
Check candidates' corresponding strain gauge data
Any pushpoint candidate experiencing force?
false
true
Save candidate experiencing highest force, as pushpoint
Visualize pushpoint

Figure 6.3: Flowchart of how pushpoints are found by using both visual and tactile data. Yellow oval marks program start up and initialization, green rectangles are processes, blue parallelograms represent input/output and red squares are system checks.

6.2.1 Handling the Obstacle Triplet Model Criteria

The essence of the obstacle triplet model is to acquire three obstacles for three pushpoints. Doing this reduces the number of dimensions in the system to one. The obstacle triplet model is detailed in [6], but is also considered here as it is a crucial step to achieve locomotion. The following bullet list contains criteria for selecting obstacle triplets

- there must be at least three obstacles for three corresponding pushpoints
- obstacle triplets must contain obstacles on alternating sides of the snake
- the distance between obstacles and joints must not exceed a certain, set limit
- first obstacle triplet to fulfill the above requirements becomes the current obstacle triplet.

Based on the bullet points the obstacle triplet model should find obstacles in a manner similar to the ones in figure 6.4 and 6.5. Obstacle 3 in figure 6.5 exceeds the distance limit criterion and is discarded from the obstacle triplet despite the fulfillment of alternating sides.

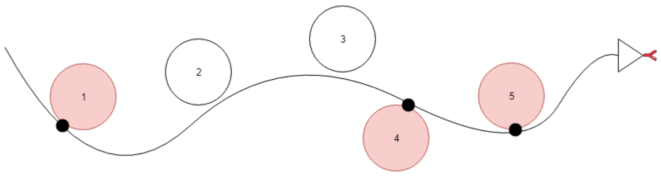


Figure 6.4: The figure shows how the triplets are chosen given the criteria of alternating obstacles mentioned in this chapter. The line between the obstacles represents the snake. An obstacle colored red marks a candidate for being used in the obstacle triplet.

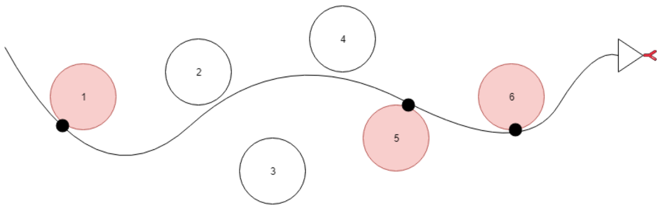


Figure 6.5: An obstacle with a distance exceeding allowed limit is introduced and discarded from the obstacle triplet. The obstacle triplet stays the same as in figure 6.4.

The first criterion, concerning the number of pushpoint candidates, is of importance based on discussions in [12], mentioned in section 2.1.1. The alternating sides issue from the list of obstacle triplet criteria, viewed in figure 6.4, is handled by first calculating the vector between the joints and obstacles. This is done by first comparing the joints' and obstacles' x- and y-positions to each other

$$\begin{aligned} x &= x_o - x_j \\ y &= y_o - y_j, \end{aligned} \tag{6.1}$$

and using the result as input for equation

$$\psi = \tan^{-1}(\frac{y}{x}). \tag{6.2}$$

Equation 6.2 cannot determine the quadrant the angle lies in with certainty because of sign ambiguity. Some extra logic is necessary and is taken care of in the program by using the built-in function $atan2()$ [27]. A simple check is performed to determine the side of which the obstacle is on based on the sign of the outputted values from equation 6.2. Obstacle side is then stored and used when assigning joints to their local obstacles.

Examination of Approaches for Obtaining Obstacle Side

Note that it is important to explicitly set maximum one pushpoint per obstacle when using equation 6.2 to find obstacle side. This is because $atan2()$ only cares about which side of the obstacle the snake is relative to the global frame, not relative to the local frame (the snake robot). This can lead to unwanted cases such as the one in figure 6.6 where the system finds two pushpoints for obstacle 2.

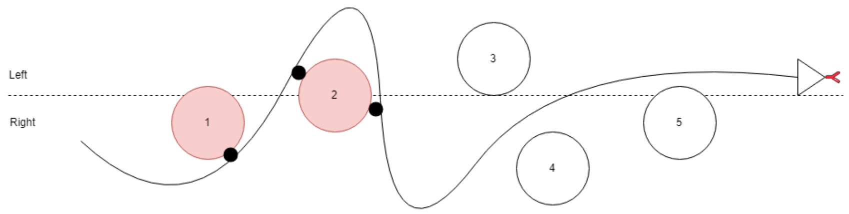


Figure 6.6: Same obstacle setup as in figure 6.4. The snake curves around obstacle 2 in a manner that register two pushpoints (Exaggerated snake posture for visual purposes). This is due to how obstacle sides are chosen. The dashed line marks the separation between left and right regarding obstacle side, from a global perspective.

Strain gauge data is, in theory, able to determine the side of the joint on which forces are acting and thus identify obstacle side. In practice joints sometimes show a normal force in the opposite direction of what the obstacle sides imply [14]. The camera system is shown to be reliable here and for this reason, the system relies on the cameras to find correct obstacle sides.

Assigning Joints to Obstacles

Obstacle sides are used in the assignment of nominating joints to their local obstacle. Assigning joints in this fashion deems it unnecessary to look for pushpoint candidates, for a particular obstacle, down the entirety of the kinematic chain. The absolute distance between $joint_i$ and $obstacle_j$ is given by equation 6.3. The limit for nominating joints to specific obstacles is defined by the added radius of both the obstacle and the joint, plus the uncertainty in position of the two, found from testing, see section 8.1. All joints inside the given limit in radius are currently being held as possible pushpoints. In

$$d(joint_i, obstacle_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{6.3}$$

where

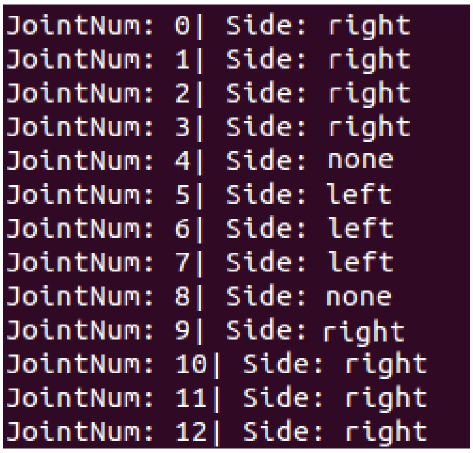
$$[0, 12] \Rightarrow \{i \in \mathbb{Z} : 0 \leq i \leq 12\}$$

and

$$[0, 2] \Rightarrow \{j \in \mathbb{Z} : 0 \leq j \leq 2\},$$

i is used for representing the joint number and j is used for the obstacle number.

Post calculations the obstacles have adopted local joints based on distance. Figure 6.7 is a cut out from a terminal window showing how the joints are organized. The elements relate to the simulated snake in figure 6.8. Joints have grouped themselves around obstacles on alternating sides, with joint 4 and 8 found to be too far away from any obstacle, indicated by getting the obstacle side *none*. Even though they fulfill the rule about alternating sides they are not recognized as pushpoint candidates. The bullet list at the beginning of section 6.2.1 is here fulfilled.



```
JointNum: 0| Side: right
JointNum: 1| Side: right
JointNum: 2| Side: right
JointNum: 3| Side: right
JointNum: 4| Side: none
JointNum: 5| Side: left
JointNum: 6| Side: left
JointNum: 7| Side: left
JointNum: 8| Side: none
JointNum: 9| Side: right
JointNum: 10| Side: right
JointNum: 11| Side: right
JointNum: 12| Side: right
```

Figure 6.7: A list of joints around their respective obstacles. The clusters of registered sides represent the grouping of local joints around the obstacles in figure 6.8. JointNum 12 in the list corresponds to the left most joint in figure 6.8.

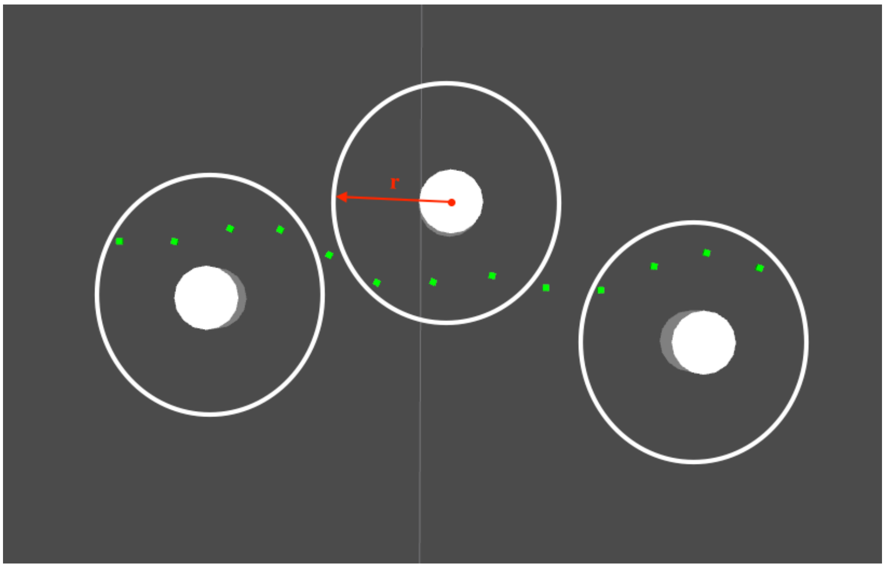


Figure 6.8: The figure shows the snake in figure 5.3 with added areas (white circles) of radius r, for showing the boundaries for what is considered local obstacle joints.

6.2.2 Locating Pushpoints

Using tactile data to verify pushpoints is the final step of the process of extracting pushpoints. This is verification by sensor fusion. For each obstacle, its corresponding pushpoint candidates are checked against strain gauge data coming from LabVIEW. The procedure requires contact between the snake and obstacles for the sensors to give out a value (in theory, due to propagating forces, forces can be registered despite being in contact). The local joint experiencing the highest force will be elected as a pushpoint for its obstacle. This is what the last check in the flowchart in figure 6.3 implies.

Figure 6.9 shows the found pushpoints after the procedure, marked as red. RViz also displays the joints closest to each obstacle, marked as blue. Pushpoints overlapping with closest joints get the color purple. Since tactile data is used to locate pushpoints, the system working perfectly should produce pushpoints that overlap with the closest joints, making them purple. Seen for pushpoints found at joint 10 and 1 (by following the snake joint numbering standard in figure 3.1) in figure 6.9. The reason being that no joint can be closer to a rigid hindrance than a joint that is colliding with it, only as close. Because the snake's joints are interconnected and their sensors are affected by one another, overlap is not always the case. Strain between joints can sometimes be bigger than the force felt for a colliding joint. This is seen in the video *ForceTest* and discussed in section 9.1.3. A general discussion about the topic of joints affecting each other in this manner is seen in section 9.4. Closest joints and pushpoints are therefore included separately, enabling the user to know when an overlap between pushpoints and closest joints occur. An example of overlap occurring for every pushpoint is found in figure 9.6.

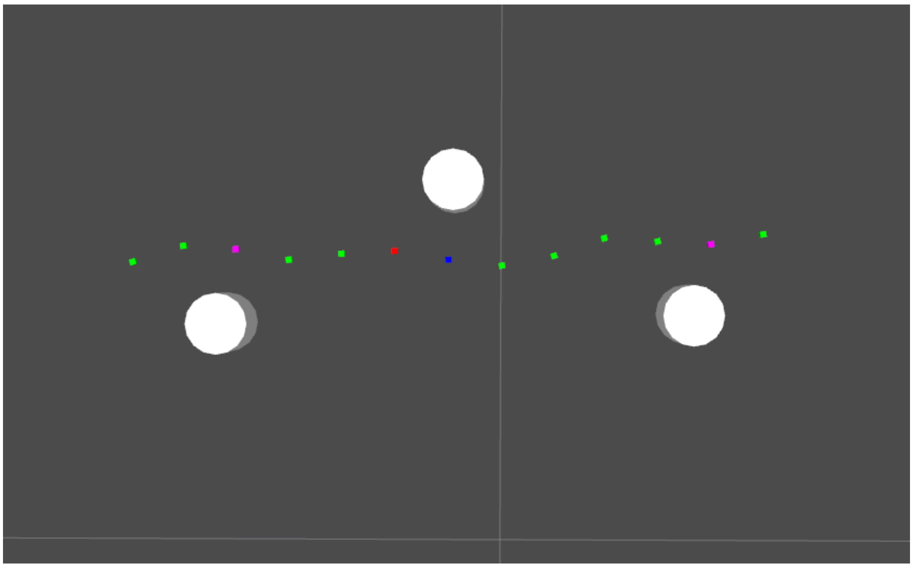


Figure 6.9: Simulated snake setup with pushpoints given by sensor fusion. Pushpoints, in red and purple, are chosen based strain gauge values. Force data is not displayed here for transparency. Joint 0 is the head of the snake and is the right most joint.

6.3 Pushpoint Selection by Visual Aid

A visual approach to pushpoint extraction, independent of tactile data, is implemented and shown in this section. Pushpoint selection by visual aid differs from selection by sensor fusion at the moment of electing pushpoints from the set of pushpoint candidates. Instead of analyzing strain gauge data for each joint to see which joint has the highest force acting upon it, a pushpoint is chosen as the foremost local joint to an obstacle. Ensuring that the chosen pushpoint always lays in front of the obstacle compared to the snake's head. This will give a resulting force pointing forwards and hence generate forward propulsion. Figure 6.10 shows the intended way of extracting pushpoints. Arrows mark the obstacle's local joints, with red arrows indicating the obstacles' connections to pushpoints. Green circles mark closest points.

This way of exploring POAL is thought to be an experiment to test how alternatives to POAL by sensor fusion do. Finding pushpoints solely by optical aid is not a substitute for sensor fusion, but may in the future be an extension. Section 10.2 talks about this idea. Section 7.7 contains the experimental setup for visual perception based testing of POAL.

Figure 6.11 shows pushpoints found by visual perception. The red squares are pushpoints, blue squares are closest joints and purple squares indicate an overlap between the previous two. Notice that the marked pushpoints are located in front of the obstacles, which is an indication of correct system behavior. Testing of POAL by visual aid

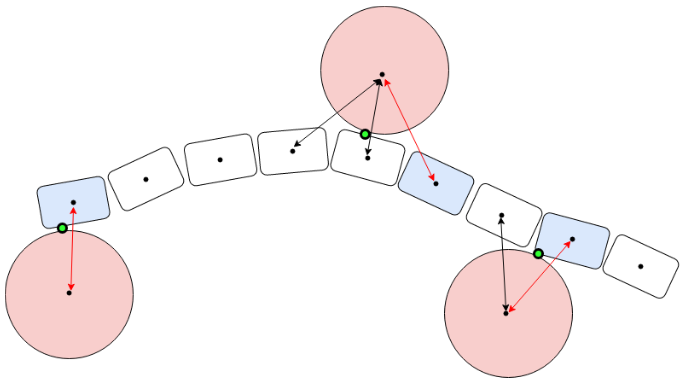


Figure 6.10: Conceptual illustration of how the camera picks out pushpoints for each of the obstacles. Choosing the foremost joint local to an obstacle as a pushpoint (marked blue). Green and black circles are contact points. Note that they not necessarily overlap with selected pushpoints.

is conducted and the results are in video form, presented in section 8.4 and discussed in section 9.1.5.

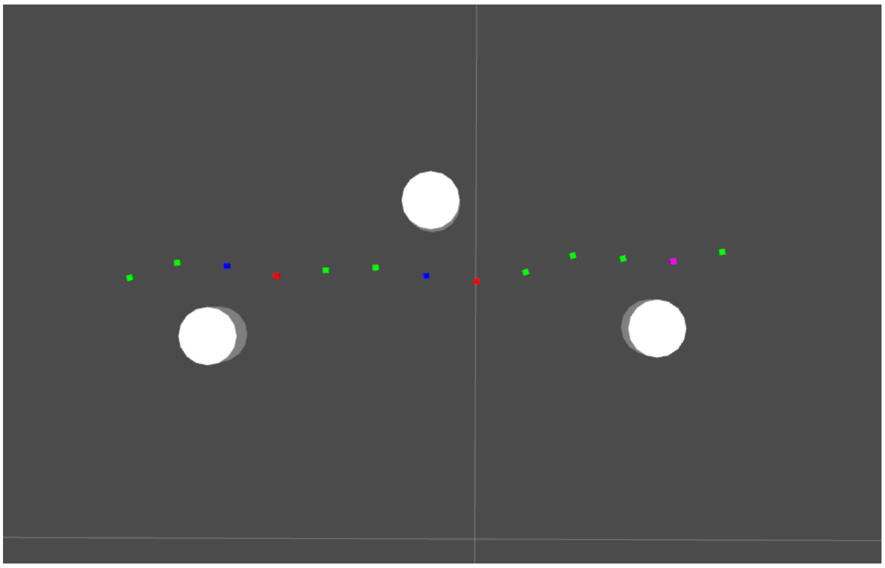


Figure 6.11: Pushpoints found by visual aid. Red squares represent pushpoints and blue squares mark the closest joints to an obstacle. Any overlap between pushpoints and closest joints will take the color purple, as seen for joint 1 following the joint numbering standard in figure 3.1.

Chapter 7

Experiments for Realization of POAL

7.1 Motivation

Experiments are conducted to demonstrate system robustness related to the implementation of the camera system and the integration of tactile data for sensor fusion. The experimental setup and methods used are described in this chapter. Successful testing marks a step towards realizing POAL with autonomous snakes.

7.1.1 Tests

Tests that are listed below in the order they are conducted

- compare the error between calculated and real joint positions based on marker configurations
- check system ability to uphold pushpoint criteria
- check the response of the simulated snake (force display in the simulated environment) when applying force to the real snake
- POAL by sensor fusion
- POAL by visual perception

Experiments regarding POAL are conducted without the use of a torque controller. The controller utilized sets desired joint angles based on desired torque. A torque controller is already implemented and detailed in [14], but further tuning is required for this to be effective.

7.2 Area of Testing

All experiments are conducted at *NTNU Gløshaugen*, room B333 (UAV laboratory). Tests are conducted on a planar surface and include obstacles of known sizes and positions, found from visual perception. Obstacles are mounted to the surface and cannot move horizontally. The testing area is well lit with both natural and artificial lighting.

7.3 Experimenting with Marker Configurations

Tests to check for deviating position values between the calculated kinematics and the visual data from Motive are conducted. The tests are executed for the purpose of verifying the preciseness of the developed kinematics calculation algorithm in combination with the snake's measured joint angles. Single markers are placed in the center of every joint, apart from joints containing rigid bodies, to track all joints visually in Motive. Monitored checkpoint joints from Motive are used as reference joints because of Motive's high level of accuracy. Figure 7.1 illustrates the different setups that are tested. The snake pose is the same for every tested configuration. Experiments also include testing of RB design (number of markers, composition, etc.), as seen in 7.1b and 7.1c. Checking if the design itself impacts tracking reliability.

Joints containing RBs have known poses and are checkpoints when determining the kinematic chain. Having many of these checkpoints may result in less deviance, but it is not desirable to have to track the entire configuration directly. Too many reference joints make it difficult for the cameras to keep track. Especially in real life scenarios like in section 10.3. In addition, it is superfluous to track every joint directly when it can be done by estimation. Finding the optimal ratio between the number of joints tracked and the number of joints calculated from kinematics is a priority. Experiments are planned with this in mind. The configuration with the lowest deviance from the reference is the one used for the implementation of the visual perception module in chapter 5. The results of the experiments are found in section 8.1.

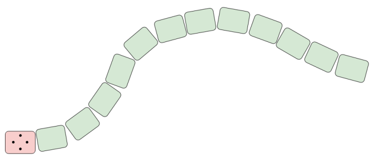
7.4 Testing Pushpoint Criteria

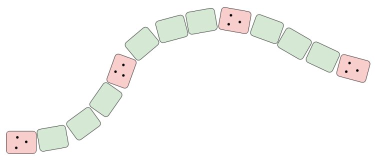
A set of tests to verify the system's ability to correctly find pushpoints is planned. Testing should reveal the system's ability to achieve the following

- find correct obstacle side relative to the snake
- know the distance between joints and obstacles
- know how many legit pushpoint candidates and pushpoints that are found and stored

Assuring the program's capability to comply with the list is crucial when trying to uphold the obstacle triplet criteria of alternating obstacle side, distance limit, etc. listed in section 6.2.1.

The setup seen in figure 5.7 is used for testing, and obstacles are observed both in RViz and directly. Obstacles are conveyed around the stationary snake to test above bullet points. The desired result is of the type illustrated in figure 6.4. Actual results, as seen in RViz, are presented in section 8.2.


(a) A single RB on the last joint (tail).


(b) Marker triplets on every four joint.

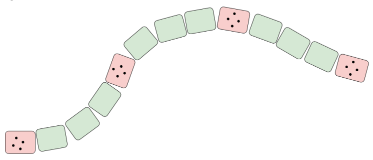

(c) Marker quadruplets on every four joint.

Figure 7.1: Tested marker configurations. Red joints represent joints containing RBs. Green joints only have single markers placed in the center. Single marker positions are written down directly from Motive's interface.

7.5 Experimenting with Force Application

Section 5.2.3 details the implementation of the visualization feature of displaying forces felt by the snake joints, in a simulated environment. It is implemented as a part of the simulation tool built in RViz. To test that forces are being displayed correctly in RViz, the joints are being exposed to forces while at rest. Figure 5.7 is the type of test setup. It is crucial for the force display in RViz to work satisfactorily when analyzing snake performance because it can reveal a lot of else hidden snake behavior.

The test is conducted by exposing the snake to an increasing force applied to three different parts of the snake. The areas of force exposure are located near the obstacles to observe strain gauge performance near obstacles. Tests are mainly done here to ensure that the simulation is displaying the correct data with regards to force magnitude, force direction, pushpoints, etc. Testing the phenomenon of propagating forces is not done here directly, but is seen in the results of the conducted test, and is discussed.

Results from testing are presented in 8.3 and discussed in 9.1. The result comes in the form of an attached video called *ForceTest*, see Appendix.

7.6 POAL by Sensor Fusion

After conducting the above tests with satisfactory results, POAL is tested. POAL by sensor fusion is tested by including tactile data (strain gauge data) with visual data, for verification. Testing is based on the validity of the results from section 7.4. Success from testing will validate the considered implementation of pushpoint extraction by sensor fusion. As mentioned earlier, the controller used sets desired joint angles based on desired torque.

The test setup is of the same type as in figure 5.7. The results from testing are presented in section 8.4 and discussed in section 9.1. The video is found in the attachments under *Videos of POAL*, called *POALTactile.mp4*, see Appendix.

7.7 POAL by Visual Aid

POAL is tested by using visual data and its corresponding logic from section 6.3. Testing is based on the validity of the results from section 7.4. POAL by visual perception is done with two different set distance limits. The distance limit is the maximum limit for accepting local obstacle joints, see section 6.2.1 for the definition. The values have a basis in the findings from testing marker configurations, see section 8.1. First, the distance limit is set to 20 cm. 20 cm represents an added uncertainty of around 7 cm. That is approximately the same error as the worst case error from testing marker configurations, see section 8.1. The distance limit is then reduced to 15 cm, which represents a much more precise marker configuration.

Success from testing will validate the selected implementation of pushpoint extraction. The test setup is of the same type as in figure 5.7. The videos from testing POAL by visual aid are found in the attachments under *Videos of POAL*, see Appendix. Other results are found in section 8.4.

Chapter 8

Results

Results from testing are presented here, in the same order as in chapter 7.

8.1 Marker Configurations

Tables 8.1, 8.2 and 8.3 compare visual positioning data with calculated data from kinematics. The three tables relate to the different marker configurations tested in section 7.3. The RMS error as it propagates down the kinematic chain is plotted in MATLAB and seen in figure 8.1 for each of the tested configurations. Large variations in RMS values are seen in the plot, with the configuration consisting of only one rigid body experiencing the highest error.

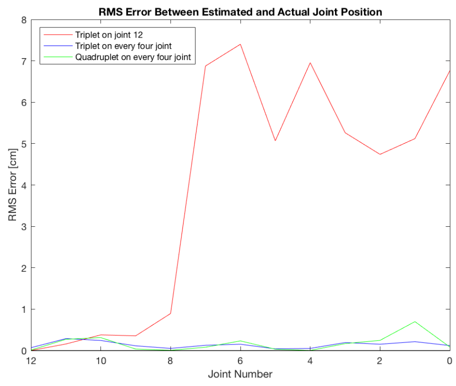


Figure 8.1: Deviation from the zero line represents propagating error between joints. Calculations start at joint 12 and move forwards.

Table 8.1: Error between calculated and actual joint positions, with marker triplet on joint 12.

joint	x calc	y calc	x OptiTrack	y OptiTrack	x error	y error	RMS error
0	1,9464	2,4695	1,9296	2,404	-0,0168	-0,0655	6,762018929
1	1,8877	2,4032	1,8451	2,4316	-0,0426	0,0284	5,119882811
2	1,7993	2,408	1,7912	2,3613	-0,0081	-0,0467	4,73972573
3	1,7422	2,3404	1,6996	2,3713	-0,0426	0,0309	5,262670425
4	1,6573	2,3654	1,6494	2,2963	-0,0079	-0,0691	6,955012581
5	1,57	2,3796	1,5631	2,3294	-0,0069	-0,0502	5,067198437
6	1,4923	2,4219	1,4765	2,3496	-0,0158	-0,0723	7,400628352
7	1,413	2,4613	1,4016	2,3935	-0,0114	-0,0678	6,875172725
8	1,329	2,4335	1,3248	2,4414	0,0042	0,0079	0,894706656
9	1,2445	2,4072	1,2426	2,4102	-0,0019	0,003	0,355105618
10	1,1599	2,3812	1,1576	2,3842	-0,0023	0,003	0,378021163
11	1,074	2,3596	1,0731	2,3609	-0,0009	0,0013	0,158113883
12	0,9882	2,3381	0,9882	2,3381	0,0000	0	0

Table 8.2: Error between calculated and actual joint positions, with marker triplets on every four joint.

joint	x calc	y calc	x OptiTrack	y OptiTrack	x error	y error	RMS error
0	1,9584	2,6972	1,9576	2,6981	-0,0008	0,0009	0,120415946
1	1,8699	2,6972	1,8680	2,6962	-0,0019	-0,0010	0,214709106
2	1,7798	2,6940	1,7795	2,6955	-0,0003	0,0015	0,152970585
3	1,6913	2,6940	1,6931	2,6947	0,0018	0,0007	0,193132079
4	1,6028	2,6940	1,6023	2,6941	-0,0005	0,0001	0,050990195
5	1,5143	2,6940	1,5139	2,6940	-0,0004	0,0000	0,04
6	1,4261	2,6931	1,4253	2,6944	-0,0008	0,0013	0,152643375
7	1,3376	2,6931	1,3367	2,6940	-0,0009	0,0009	0,127279221
8	1,2491	2,6931	1,2486	2,6932	-0,0005	0,0001	0,050990195
9	1,1606	2,6931	1,1595	2,6928	-0,0011	-0,0003	0,114017543
10	1,0735	2,6940	1,0712	2,6948	-0,0023	0,0008	0,243515913
11	0,9850	2,6940	0,9834	2,6916	-0,0016	-0,0024	0,288444102
12	0,8965	2,6940	0,8959	2,6941	-0,0006	0,0001	0,060827625

Table 8.3: Error between calculated and actual joint positions, with marker quadruplets on every four joint.

joint	x calc	y calc	x OptiTrack	y OptiTrack	x error	y error	RMS error
0	1,7915	2,4961	1,7915	2,4969	0,0000	0,0009	0,085
1	1,7415	2,4230	1,7360	2,4273	-0,0055	0,0043	0,698928466
2	1,6691	2,3681	1,6685	2,3705	-0,0006	0,0024	0,245489307
3	1,5917	2,3253	1,5914	2,3269	-0,0003	0,0017	0,168074388
4	1,5081	2,2963	1,5081	2,2963	0,0000	0,0000	0,001414214
5	1,4209	2,2813	1,4206	2,2811	-0,0003	-0,0002	0,031906112
6	1,3320	2,2803	1,3319	2,2826	-0,0001	0,0023	0,233308808
7	1,2450	2,2962	1,2448	2,2970	-0,0002	0,0008	0,079831072
8	1,1620	2,3269	1,1619	2,3269	-0,0001	0,0000	0,00509902
9	1,0860	2,3723	1,0862	2,3720	0,0002	-0,0003	0,036055513
10	1,0243	2,4340	1,0218	2,4322	-0,0025	-0,0018	0,312833822
11	0,9759	2,5081	0,9741	2,5061	-0,0018	-0,0020	0,271309786
12	0,9451	2,5911	0,9451	2,5910	0,0000	-0,0001	0,01

8.2 Pushpoint Criteria

The presentation of experimental results regarding the system's ability to comply with given pushpoint criteria follows in this section. Experimental setup is described in section 7.4 and a discussion of the topic is found in section 9.1.2. As stated earlier the snake joint numbering standard used is the one found in figure 3.1. Figure 8.2 shows two pushpoints found for joint 12 and 1 respectively (marked blue), as a result of the leftmost obstacle being moved to the same side as the middle one. Figure 8.3 shows the system finding candidates at joint 6 and 1 respectively. The leftmost obstacle is placed too far away from the snake and no pushpoint is found here. Moving the left most obstacle in figure 8.3 a little bit closer to the snake results in the snake in figure 8.4. Three pushpoints are found in this case.

8.3 Results from Experimenting with Force Application

Results from testing the application of forces on joints are presented in the video *ForceTest.mp4* in the attachments. White arrows are normal forces and green arrows are tangent forces in the video. Figures of RViz in section 9.1.3 are snapshots from the test video. The video shows three distinct intervals of pressure applied to three different parts of the snake. One for each obstacle. RViz runs parallel to the video of the real snake setup and reactions are seen immediately. Meaning that the real snake is viewed side by side with the simulated snake. This is also the case for the results in section 8.4.

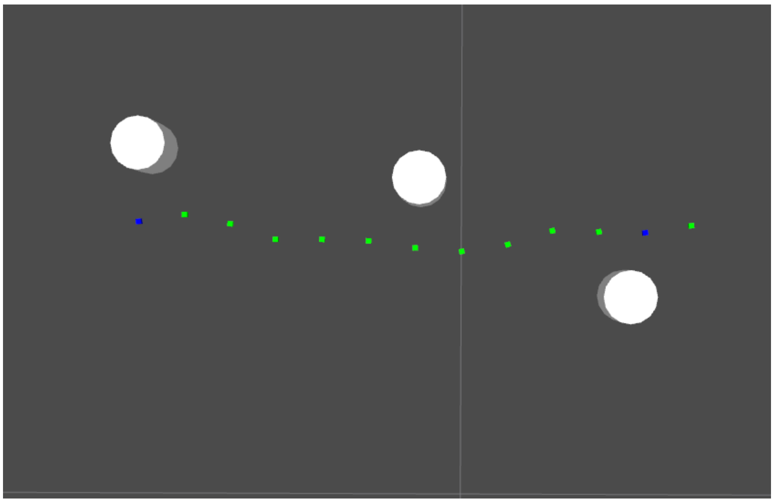


Figure 8.2: A picture of the simulation not picking up on the second pushpoints, ergo failing the criterion of alternating sides. The snake moves from left to right.

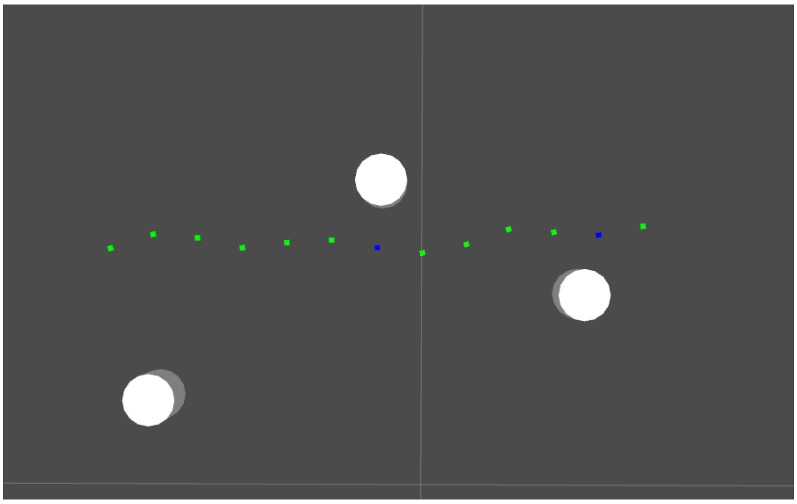


Figure 8.3: Two pushpoints found. The last obstacle is placed too far away from the obstacle, hence failing the pushpoint criterion of maximum distance.

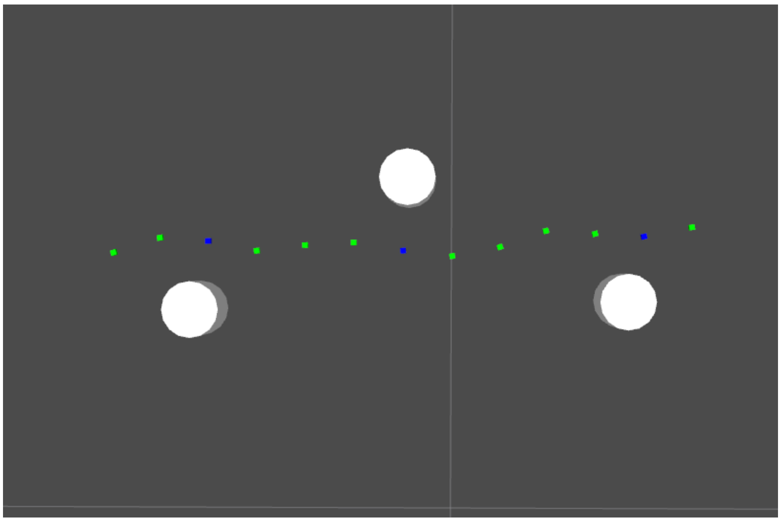


Figure 8.4: Three pushpoints are found when upholding the alternating sides and relative distance criteria.

8.4 Results from Testing POAL

Results from testing POAL with both sensor fusion and with only the use of visual perception are presented in this section. Snake setups and executions are filmed and included in the attachments. A link in section 11.1.1 in Appendix redirects to the same videos. These videos are the main results from testing. Videos are without displaying force arrows. This is done for transparency. Pushpoint verification by changing joint color is deemed sufficient for understanding the behavior.

Figure 8.5 shows the snake moving towards a target, starting at roughly 45 cm away and ending up approximately 20 cm away from it. Spikes appear in the plot with inconsistent intervals.

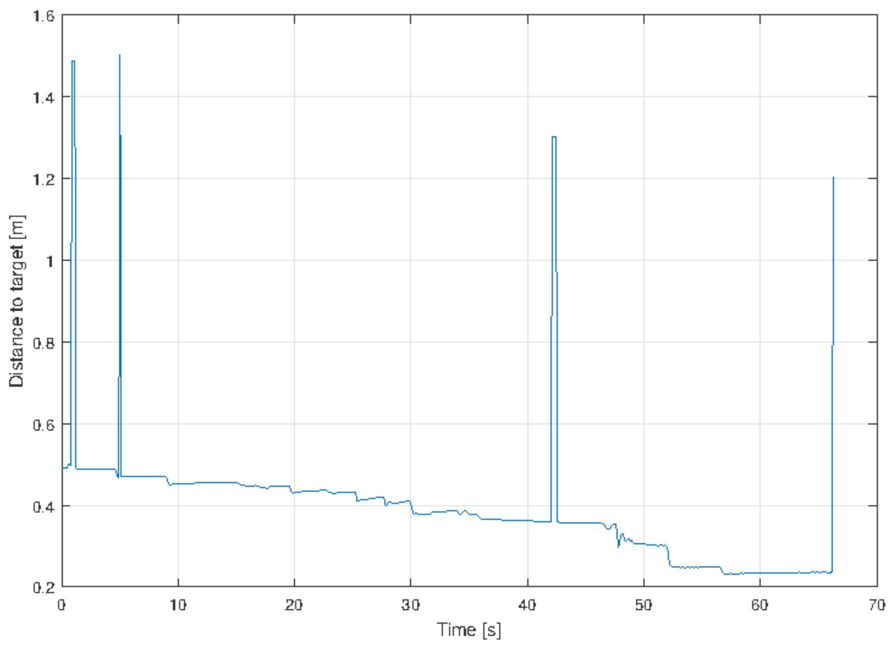


Figure 8.5: Snake closing in on a predefined target. Substantial spikes in distance occur at uneven intervals.

Chapter 9

Discussion

This chapter first discusses the experimental results, in section 9.1, then the project in general in the other sections.

9.1 Discussing Results from Testing

Discussions in light of results from laboratory experiments follow in this section, in the same order as they appear in chapter 7 and 8.

9.1.1 Experimenting with Marker Configurations

Table 8.1 shows RMS errors almost reaching 7.5 cm for joint 6 with the use of a single rigid body. As figure 8.1 shows, a huge discrepancy appears between the seventh and eight joint for the configuration with one RB at the tail. It may be due to wrong outputted servo data caused by the lack of sufficient sensor calibration or malfunctioning hardware. See a relevant discussion of sensor calibration in section 9.5.

For transformations between joints further up the chain (joint 7 - joint 0) the error variance is pretty stationary compared to the huge error between joint 8 and joint 7. This also helps backing up that the spike is an abnormality. Apart from the rapid increase in error seen in the plot, the marker configuration consisting of only one RB follows the reference snake pretty well. The problem with using only one RB is that this type of configuration has no more checkpoint joints along the snake. The error is therefore not adjusted for, and thus persists. This is risky and before measures are taken to ensure correct performance for the use of only one RB, this is not an option. Note that calculations start at joint 12 and move forwards.

Interpretation of figure 8.1 reveals an advantageous strategy of using multiple marker configurations. Both marker configurations containing several RBs achieved an RMS error of under 0.7 cm at the most. Their matching behavior indicates non-relevancy regarding the design of the RBs, assuming that the value of joint 1 in table 8.3 can be discarded as an irregularity. Such irregularities can be caused by jumps in RBs' orientation in Motive due to light pollution or the relative movement of RB markers. In the case of joint 1 in table 8.3, it is reasonable to believe that joint 0 experienced some sort of error. Since joint 0 is a checkpoint joint it is supposed to be precise in its position and used for calculating the positions of joint 1 and 2. Compared to the other checkpoint joints in the table (joints 4, 8 and 12) joint 0 has many times the deviation and could be the reason for the huge discrepancy in position for joint 1. The position of joint 2 is calculated as the mean of joint 0 and 4. Joint 4 is seen from the table as very precise and hence corrects the error of joint 2 to a degree. See chapter 5.1.1 for the algorithm to calculate the kinematic chain.

The configuration implemented in the end is a mix of the best configurations based on RMS error in figure 8.1. The reason for mixing them is Motive's tendencies to switch around RBs of identical design. Creating a set of RBs with different constellations will prevent Motive from switching the RBs around when the RBs are exposed to too much light pollution or when RBs are hiding behind obstacles.

Using one RB instead of four is an easier task when it comes to tracking. Experiments regarding marker configurations could have been more numerous, trying to figure out what went wrong when equipping the snake with only one rigid body. Solving this may have resulted in a system dependent on only one rigid body. A solution was not found as it was not deemed necessary to spend too much time dwelling on it. The system works for four tracked joints and nothing prohibits future versions in changing the number of tracked joints. It is then recommended to use more precise hardware (or better calibrated hardware).

9.1.2 Testing Pushpoint Criteria

From the plot in figure 8.2 it is revealed that the system is able to handle the issue of same side obstacles. The algorithm starts at the obstacle closest to the snake's tail and advances towards the head. The first obstacle is found to be on a valid side because finding the first pushpoint does not require a specific obstacle side. Since the first obstacle is located to the *left* of the snake, the next valid obstacle side is *right*. Consequentially the obstacle in the middle is discarded. The rightmost obstacle is on the other side of the snake compared to the leftmost obstacle and is approved.

The plot in figure 8.3 shows the alternating sides issue being taken care of, but the pushpoint candidate is still being discarded because its distance to the nearest obstacle exceeds the set distance limit. Straightening itself out would make the snake gain a third pushpoint, giving the snake the sufficient number of pushpoints needed for motion. The snake has no way of knowing the existence of the nearby obstacle based solely on tactile data and is left with the illusion of not having enough nearby obstacles to be able to move forward. Handling these kinds of situations is what makes for a robust system and is a part of what encouraged the implementation of the visual perception module in this project.

Both of the above scenarios end with an insufficient amount of pushpoints. As a result, the snake is not able to move forward. These tests illustrate correct system behavior regarding pushpoint extraction. Figure 8.4 shows a scenario where the distant obstacle from figure 8.3 is moved closer to the snake. The alternating sides and distance criteria are as a consequence upheld, resulting in a sufficient amount of pushpoints for motion.

Note that the figures from testing pushpoint criteria display *closest joints* and not *pushpoint joints*. The reason is that the development of the pushpoint extraction algorithm in figure 6.3 is done after assuring that the system upholds all pushpoint criteria with the use of closest joints. Pushpoints are extracted from the same set of local obstacle joints as closest joints are. Meaning that successful results from testing with closest joints also incorporate successful results with pushpoints.

9.1.3 Experimenting with Force Application

Video *ForceTest.mp4* in the attachments shows how the snake reacts to forces applied to different places on the snake. RViz is here shown to be a precise tool for investigating which joints are involved in a collision. Figures 9.1 and 9.2 contain snapshots from an RViz perspective in the video, *ForceTest.mp4*. Note that the used pushpoint extraction algorithm used in the force experiments is the one based on sensor fusion. Strain gauge data is therefore essential in determining pushpoints.

It is seen for the middle obstacle in figure 9.1 that the system selects the local joint with the highest experienced force as a pushpoint, detailed in section 6.2.2. The system does so with no regard for force direction. This is due to the phenomenon mention in section 6.2.1 where the strain gauges sometimes register a force in the opposite direction of the actual force direction. This is an important reason for why the system should not rely solely on tactile data, but also incorporate data from the camera system. Instead of relying on the force direction outputted from the tactile module, the system uses the obstacle sides, found by utilizing the camera module, in combination with the force magnitude found from the tactile sensory. Pushpoints are being validated using this information together with the set pushpoint criteria.

Another snapshot from testing is shown in figure 9.2. A force is applied to joint 5, and the figure shows how the force propagates up the kinematic chain, towards the head. This is because of the strain created between the joints. Propagated forces due to strain between joints are highly interesting because they can cause situations where pushpoints found by sensor fusion are not even in contact with an obstacle. But because the strain force is so large, it exceeds the forces generated by collisions between the robotic snake and an obstacle. This incident is shown in figure 9.3. Notice that the rightmost joint in RViz is colored red, indicating that it is a pushpoint. Notice also that the corresponding real-time picture of the real snake reveals that the mentioned joint, selected as a pushpoint, is not in contact with the obstacle. A hypothesis is that the friction from the surface is enough to establish a strain force between the two rightmost joints, bigger than the collision force. This behavior is unfortunate and should be taken care of. For instance by changing the type of sensory to match the snake in Gazebo, see section 9.4.

Even though the force applying instrument is pressing against joint 1 in figure 9.3, force magnitudes for that same joint in figure 9.2 are practically zero. The instrument is mostly just pushing the joint casing against the obstacle so the sensor is not affected here. Implementing a *bumper* sensor could change this lack of force display, see section 9.4.

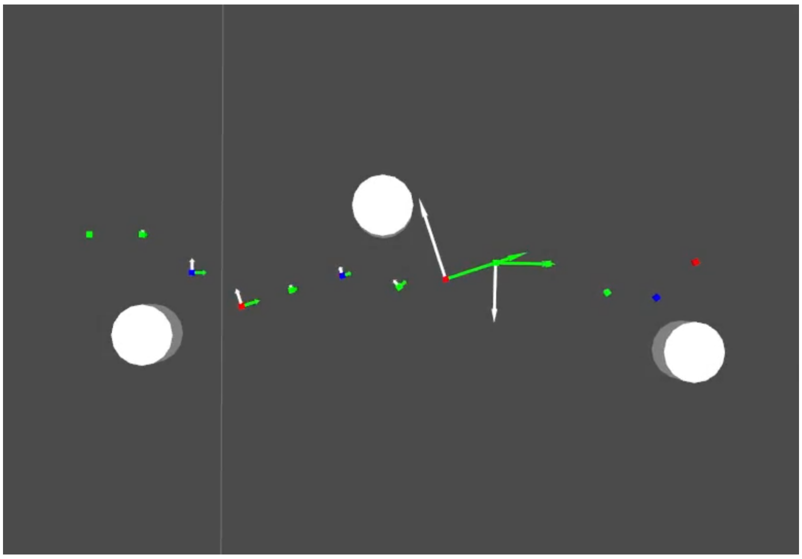


Figure 9.1: Snapshot from the video, *ForceTest.mp4*, displaying a force applied to the snake, in RViz. The middle pushpoint (marked red) experiences the highest force and is thus chosen as a pushpoint. Regardless of force direction.

9.1.4 A Sensor Fusion Approach to Achieve POAL

Working POAL

As seen in the attached video *POALTactile*, the sensor fusion approach of finding pushpoints seems to work satisfactory. Pushpoints are at times switched around frequently. The reason for this happening has to do with pushpoints being found by electing the one experiencing the highest force. As the snake moves, the forces experienced by the joints vary. It is not always the case that the joint feeling the greatest force is the most desirable joint to be elected pushpoint. Strain between joints can cause forces greater than those generated by colliding with obstacles. Forces displayed in figure 9.3 illustrates the phenomenon.

A look at the real snake's associated simulation in the video *POALTactile.mp4* reveals that the joints from time to time experience sudden displacements in RViz. This has

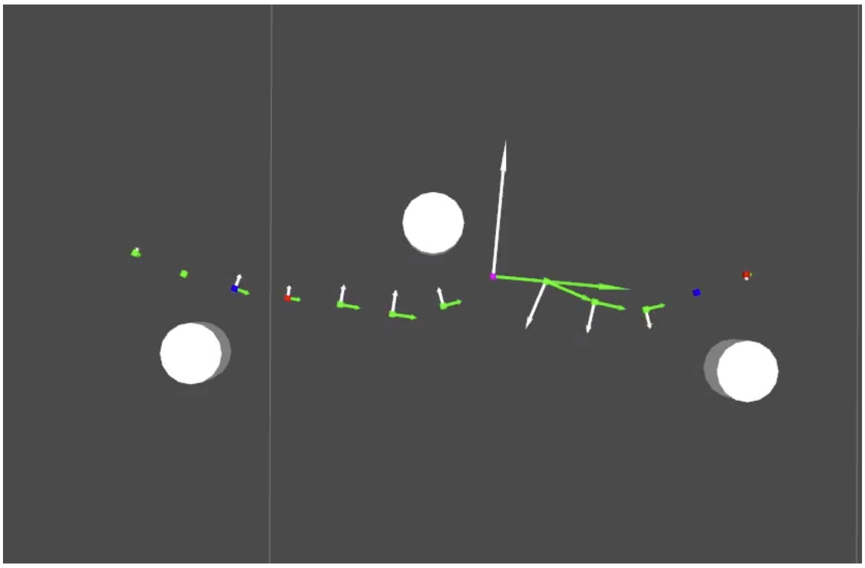


Figure 9.2: Snapshot from testing, showing force propagation between snake joints. Joints $5 - 2$ are each seen absorbing some of the force, reducing it for the next joint.

most likely to do with the camera system's tracking abilities, discussed earlier in section 9.1.1. Luckily the rapid movements last only a short while and do not seem to influence the system in a major way.

The way the propulsion works is by applying a torque to the middle pushpoint, contracting the snake in a manner that pushes the back part of the snake forwards. See figure 9.4 for snapshots of this happening, from testing POAL by sensor fusion. The controller used is based on setting a desired angle, which the middle pushpoint is trying to reach. Note that the snake contracts until desired joint angle is concluded, then rapidly straightens out. [14] details this way of motion and describes this as spring behavior. The spring behavior is seen in all attached videos regarding POAL.

As figure 9.4 shows, the propulsion relies on the middle pushpoint joint laying in front of the obstacle. Because of the discussion in this section's first paragraph regarding strain between joints, unfortunate episodes, causing the reversing of the snake, played during testing of POAL by sensor fusion. See the discussion in the following.

Reversing Snake

Including tactile data for ensuring that chosen pushpoints are in contact with an obstacle reveals a possible flaw in the current system. An incident occurred during testing where the second pushpoint was located behind an obstacle relative to the snake head. Applying a torque here caused the snake to move backwards. See figure 9.5 for a reenactment of the scenario. Associated simulation of the snake in RViz, with found pushpoints (marked

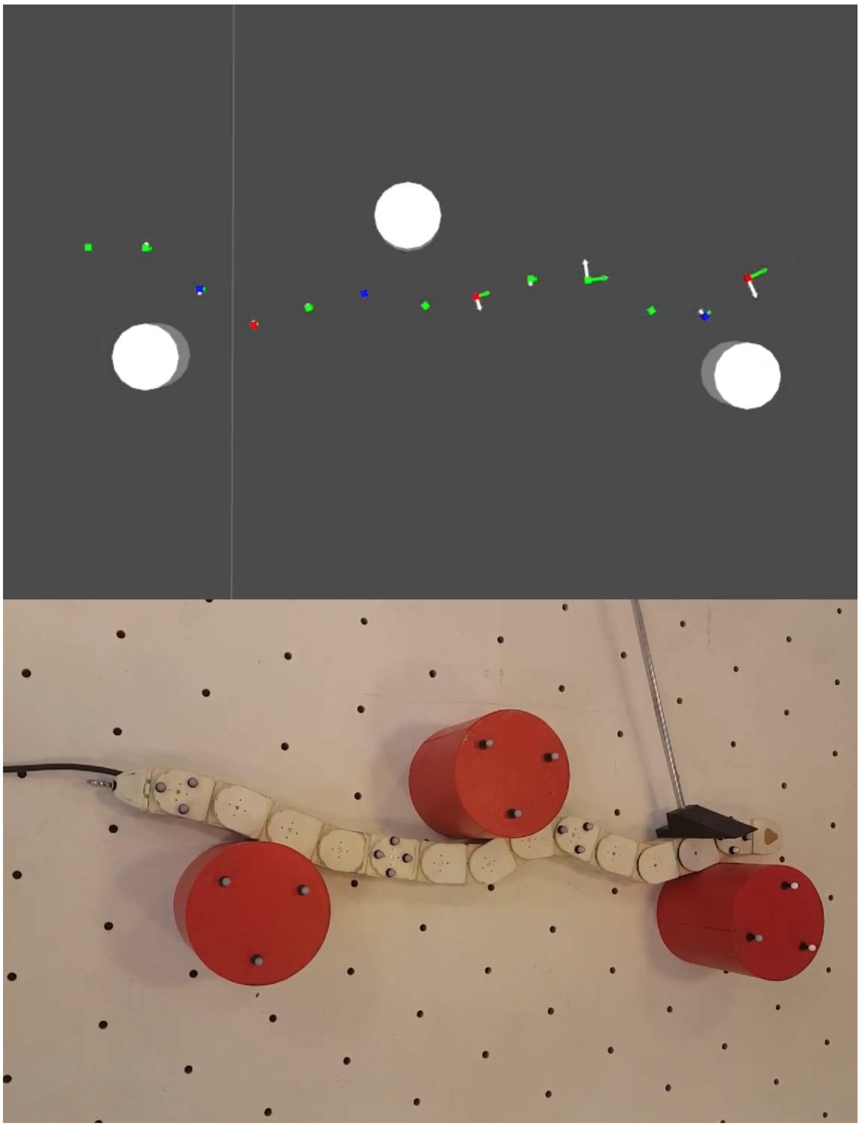
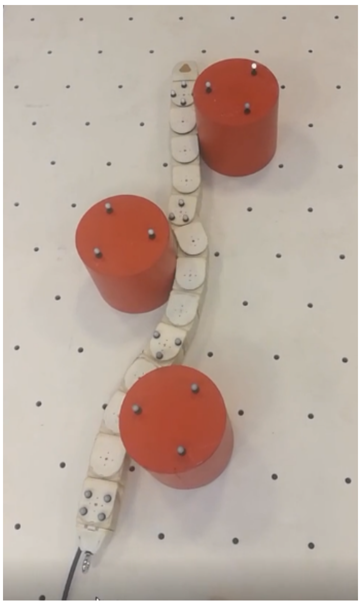
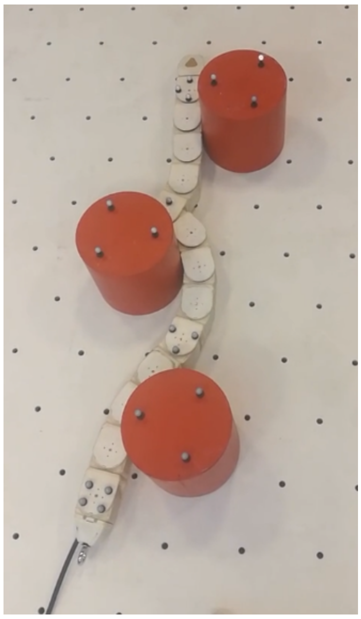


Figure 9.3: Snapshot from the video *ForceTest* showing the rightmost joint experiencing the highest force for its current obstacle, and hence elected pushpoint. The real image reveals that the pushpoint is not touching the obstacle.



(a) Snake pose before applying torque to the middle pushpoint (joint 4).



(b) Snake pose after torque is applied to the middle pushpoint (joint 4).

Figure 9.4: Snapshots from testing showing how the snake propels itself forward. The pictures are taken 4 seconds apart, with the left picture being the first snapshot taken.

as purple), is shown in figure 9.6. Pushpoints and closest joints overlap here, seen by them turning purple. The overlap indicates that the closest joints are in fact the ones experiencing the highest force. Hence, the correct pushpoints as far as the the current implemented algorithm is concerned, are found. For reference, the same snake setup as in figure 9.5 is visualized without sensor fusion in figure 9.7. A look at the found pushpoints here shows the location of the middle pushpoint being in front of the obstacle. Applying torque to the middle pushpoint here will make the snake move forwards.

These kinds of unwanted situations can be handled by checking that the sum of tangent forces points in the desired direction. The visual perception module keeps track of all joints local to an obstacle. If the joint experiencing the highest force has an undesired position, another more suitable pushpoint can be elected from the set of local joints.

Figure 8.5 plots the snake's distance to a specified target. Tall spikes in the plot appear in uneven intervals. It seems OptiTrack is having trouble keeping track of RBs when in motion. It may be the testing area that is exposed to too much light or wrongly calibrated cameras that make the RBs switch places in Motive. A look at figure 5.7 reveals that the snake used for testing has identical marker configurations for its head and tail. Section 9.1.1 says that each RB should have a unique design. In practice this is not possible because of limitations in joint surface area. Thus two triplets and two quadruplets are used to minimize the chances of RBs getting switched around. The

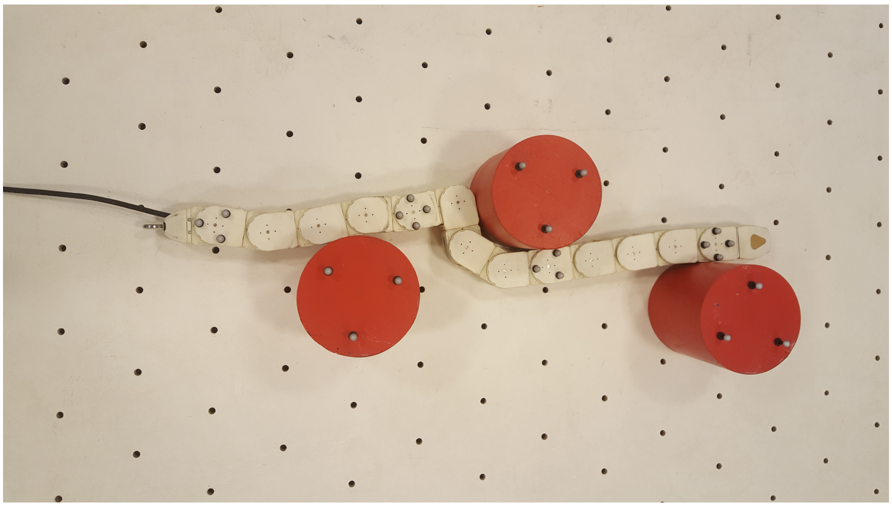


Figure 9.5: A reenactment of the scenario of a reversing snake. The highest force experienced by a joint local to the middle obstacle is felt by joint 7 following the joint numbering standard in figure 3.1. The joint then tries to reach a desired angle, which results in the snake reversing.

snake is approximately 1 meter long when moving between obstacles. The spikes seen in the plot of figure 8.5 are in the same range lengthwise. This fact backs up the claim that the spikes appear due to RBs at each end of the snake (which both are quadruplets) are switched around. To eliminate this behavior the rigid body designs have to be unique. It should be looked into for future work.

9.1.5 A Visual Approach to Achieve POAL

Section 6.3 describes the method of finding pushpoints using visual aid by setting the foremost local pushpoint candidate as the current pushpoint. In section 6.2.1 it is stated that the distance limit for pushpoint selection is the added radius of an obstacle and a joint, plus a value of uncertainty. The value of uncertainty is based on the error between actual and estimated positions found from experimental results, section 8.1. Testing was done with a distance limit first set to 20 cm and then reduced to 15 cm, see experimental setup in section 7.7. Based on joint and obstacle specifications in section 3.1, the 20 cm distance limit set, corresponds to an added uncertainty error of approximately 6.5 cm. Which approximates the worst case joint position error found in section 8.1. The video *POALVisualDistantPushpoints.mp4* represents the case where the error between actual and estimated joint positions are of the worst case presented in figure 8.1. The largest error in figure 8.1 is almost as large as a link length. The consequences of this is discussed in the following.

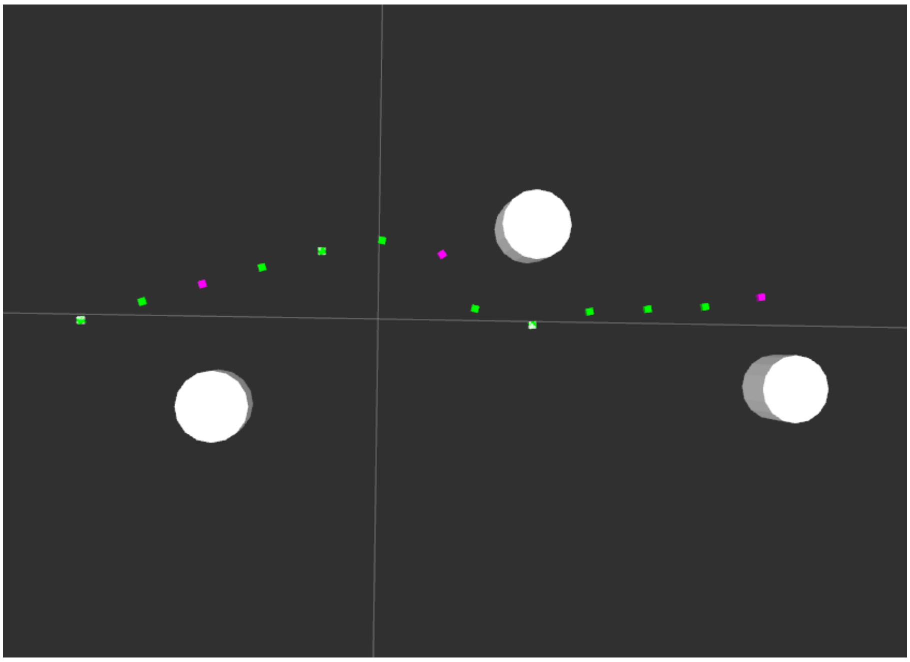


Figure 9.6: Pushpoints found by sensor fusion in the case of the snake configuration in figure 9.5.

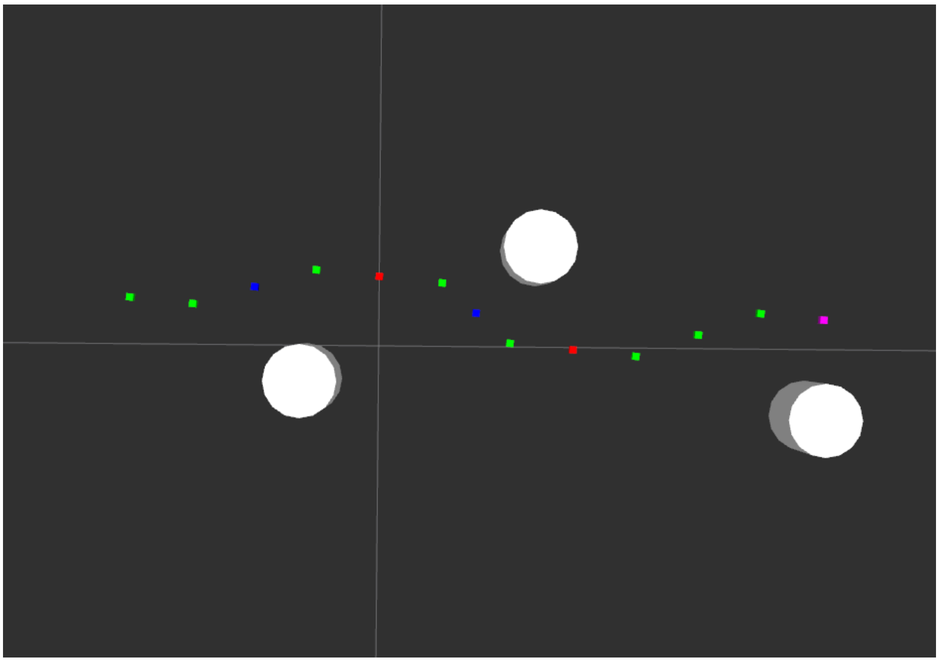


Figure 9.7: Pushpoints found by visual data in the case of the snake configuration in figure 9.5.

Excessive Distance Limit

The scenario of selecting pushpoints too far away from an obstacle demonstrated in the video *POALVisualDistantPushpoints.mp4* shows the selected pushpoint joint generating a torque, but because of the joint's distance away from the obstacle, no push is generated. Even though desired joint angle is reached, it barely touches the obstacle, creating no forward motion. Consequentially the snake remains stationary, resulting in the program not electing a pushpoint further back in the chain. When desired angle is produced, the snake has orders to straighten itself out, before setting a new desired angle. The act of straitening out makes the snake move a bit in a direction. This pseudo-random movement eventually enables the snake to move far enough in a direction to elect a more suitable pushpoint. It is a slow process and not at all desirable. Reducing the distance limit is tried.

Reducing Distance Limit

Solving the problem viewed in *POALVisualDistantPushpoints.mp4* requires a smaller distance limit for assigning local joints to obstacles. Video *POALVisualReducedPushpointDistance.mp4* demonstrates how a more accurate calculated distance limit of 15 cm yields a greater push and thus faster locomotion. The video shows the snake joint reaching its desired angle, pushing against the front of the obstacle before stretching out, leaving the snake further ahead than before. The benefit of utilizing cameras and the current algorithm is that it is very consistent in finding correct pushpoints given an appropriate distance limit, as indicated by the results.

The current way of utilizing visual perception to find pushpoints is only applicable for known obstacle sizes and positions. The reason being that this kind of information is needed to set the distance limit. This thesis is for laboratory purposes only and new methods must be considered before a functioning system outside the controlled testing area is to be realized.

9.2 Necessary Geometric Condition for Achieving POAL

A geometrical analysis is conducted in [6] given a particular obstacle triplet model. Conditions like the one in figure 9.8 are not ideal for forward motion. No propulsion is produced in this case. A few assumptions are considered in the paper for the statement to be valid. The obstacles are denoted o_1, o_2, o_3, the radii of the circle are denoted r_1, r_2, r_3 and f_1, f_2, f_3 are normal forces. There is also a tensile force acting, the tangent force of o_1. Extending the lines in the direction of the normal forces, following the paths of the radii, reveals a focus point in the center of the arc, o_c. The radii are zero with respect to o_c and the normal forces will therefore not produce any propulsion. Resulting torque balance around o_c yields

$$f_s \times r_1 = 0 \Leftrightarrow f_s = 0 \forall r_1, r_2, r_3. \tag{9.1}$$

Since the tensile force f_s is null, no propulsion force is produced. According to [6] this has relevancy in many application. For example in stability problems, efficiency optimization, pre-configuration or reconfiguration operations and control allocation operations. Measures should be taken to decrease or eliminate the chances of ending up in these particular configurations. Solutions to this problem is outside the scope of this thesis.

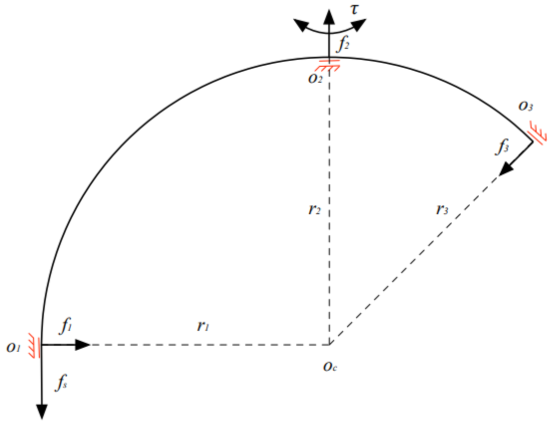


Figure 9.8: Snake described as an arc. Normal forces all intersect in the center, preventing any propulsion force of being produced because of this singularity. The illustration is retrieved from [6].

9.3 Disappearing Markers

An undesirable scenario is when one or several of the tracked RBs are lost for the cameras or switch places as discussed for the results from testing POAL, section 9.1.4. There are multiple reasons for why this can occur. Examples being light pollution or joints disappearing behind obstacles. Measures to prevent this of being an issue should be taken. Section 10.1 proposes a solution.

9.3.1 Light Pollution

The current testing location suffers from heavy light pollution on sunny days due to the room having several windows on multiple sides. Non-sufficient covered windows lead to the camera system registering undesired reflective points in the area of testing. The new marker registrations cause RBs to occasionally switch places, ending up far away from their actual positions, like in figure 8.5. Also, the deviation between estimated marker positions and actual marker positions, described in section 3.2.2, seems to occur more rapidly as light intensity increases. To conduct more reliable testing over longer periods of time it is recommended to continue testing in an environment void of external light.

The joints are not coated with non-reflective materials. Coating them can help tracking by increasing the contrast between the markers and the joints.

9.4 Force Propagation between Joints

If a joint along the snake experiences a torque, force sensors all over the snake are triggered, as indicated by the results from testing. As stated earlier, this phenomenon originates from the way the snake is interlinked. The use of strain gauge sensors causes links further up or down the chain to react because forces propagate through the snake. This is seen in figure 5.6 where forces, displayed as arrows, are registered in joints far away from the actual contact point. It is also seen in the attached video, *ForceTest.mp4*, discussed in 9.1.3. Figures in section 9.1.3 are snapshots from the video. They help illustrate the force propagation issue.

The snake simulated in Gazebo is equipped with another type of sensors called *bumper* sensors. These sensors are located in the snake links, versus in the snake joints for the real snake, and detect forces more straight to the joint. Much like how a button detects force. Consequentially, torque applied to adjacent joints does not affect other bumper sensors. The use of this kind of sensors will eliminate the issue of strain between joints exceeds the force magnitude from collision forces. As a result, it will then make the system less reliable on external observers because the system can use the snakes on-board sensory to sense obstacle sides.

The visualization of forces in figure 5.6 is part of the developed system for monitoring snake behavior. With the help of the developed simulation software, the phenomenon of force propagation is clearly displayed. It is an indication of this software's potential for future use of detecting abnormalities.

9.5 System Pre-Calibration

Measured joint angles in LabVIEW drift over time and need to be reset now and then. When resetting the measured angles to zero all joints have to lay straight, with zero heading (yaw). The same goes for Motive joints when resetting heading (yaw) for the rigid bodies. Calculating the kinematic chain is based on knowing the exact orientation of the joints. If pre-calibration is not done with sufficient accuracy regarding joint orientation, the snake can end up looking like in figure 5.4c, or worse. Here, the calculation of the snake pose has used the wrong joint orientations and positions differ more than they should.

Chapter 10

Future Work

10.1 Handling Disappearing Markers

The snake traverses the horizontal plane and collides with sturdy obstacles that are stretching 20 cm into the third dimension. The setup is, as described earlier, monitored by the utilization of 16 OptiTrack cameras, see section 3.2. Despite the number of observed angles situations where RBs, marking a snake joint, disappear from view can occur when traversing a cluttered field.

The ROS node *snakebot_kinematics* keeps track of all the links in the kinematic chain. The kinematic chain relies on visually tracked joints by a camera system that is as of now tracking several joints spread out along the snake. If just one of the tracked joints is interrupted, the basis on which the kinematic module rely is compromised for that region of the snake. This is seen in the video *POALTactile* where joints experience sudden movements in position.

A proposed solution may lie in creating an adaptive system that tracks as many joints as possible but only needs one joint to be tracked to work properly. Changing reference joint, for calculating the kinematic chain, depending on which joints are available for the cameras. If for instance only the tail joint is available, the kinematics will use this as a basis for calculations, see figure 7.1a. If more joints are available the basis could be as in figure 8.2 and so on.

10.2 Pushpoint Extraction by On-board Visual Aid

Operating a robotic snake outside the laboratory environment requires another pushpoint extraction practice. As mentioned in section 2.1, [16] strives to map a robot's surroundings by using on-board visual aid. Not relying on external sensing can increase the level of autonomous capabilities in demanding areas.

Adopting the robot's map-making abilities to be used by a snake can give it the ability to find surrounding pushpoints. The map generated can be compared to the calculated positions of the snakes joints. The snake can then know the relative position between itself and its surroundings. This knowledge can again make the snake gain information about possible nearby pushpoints without relying on actual changes in strain gauge data.

This idea is taking the concept of finding pushpoints by visual aid in section 6.3 a step further. Instead of using cameras as external observers, the snake uses a map of previously processed images as the external observer. This technique assumes that the environment the snake is traversing is relatively immutable so to be certain that the possible pushpoints persist over time. The use of computer vision in combination with a machine learning algorithms can classify desired pushpoints along the mapped areas and train the snake to know which pushpoints it should pursue.

10.3 Collaborating Robots

Continuous advancements in the field of robotics make future operations with the use of multiple robots possible. Swarm technology has a lot of benefits. Examples of collaboration between robots follow.

10.3.1 Drones Aiding Snakes in Search and Rescue

Advancements in the field of robotics open up for the possibility of cooperation between multiple robotic units, [28]. It can be envisioned a collaboration between drones and robotic snakes where drones transport the snakes to required areas. Places to dangerous or unreachable for humans. The snake can be designed in a modular manner. Having the ability to split into smaller modules to cover a greater area. After drop off the drones can act as a relay between a base and the robotic snakes, for message transmissions. Relays can also be deployed by the snakes traversing the subterranean. Covering communication even in deep underground tunnels.

Figure 10.1 illustrates the mentioned scenario where drones deploy modular, robotic snakes, equipped with transceivers, to investigate a region. The simplified drawing depicts how the snake splits up into smaller parts to cover more ground. Relays are left behind as the snake advances through the unexplored, ensuring connection between the different system modules. The drone also functions as a relay, redistributing messages between system organs. The idea of multiple entities collaborating is motivation for tackling important tasks like internal communication. Mesh networks as a means of communication are talked about in section 10.4.

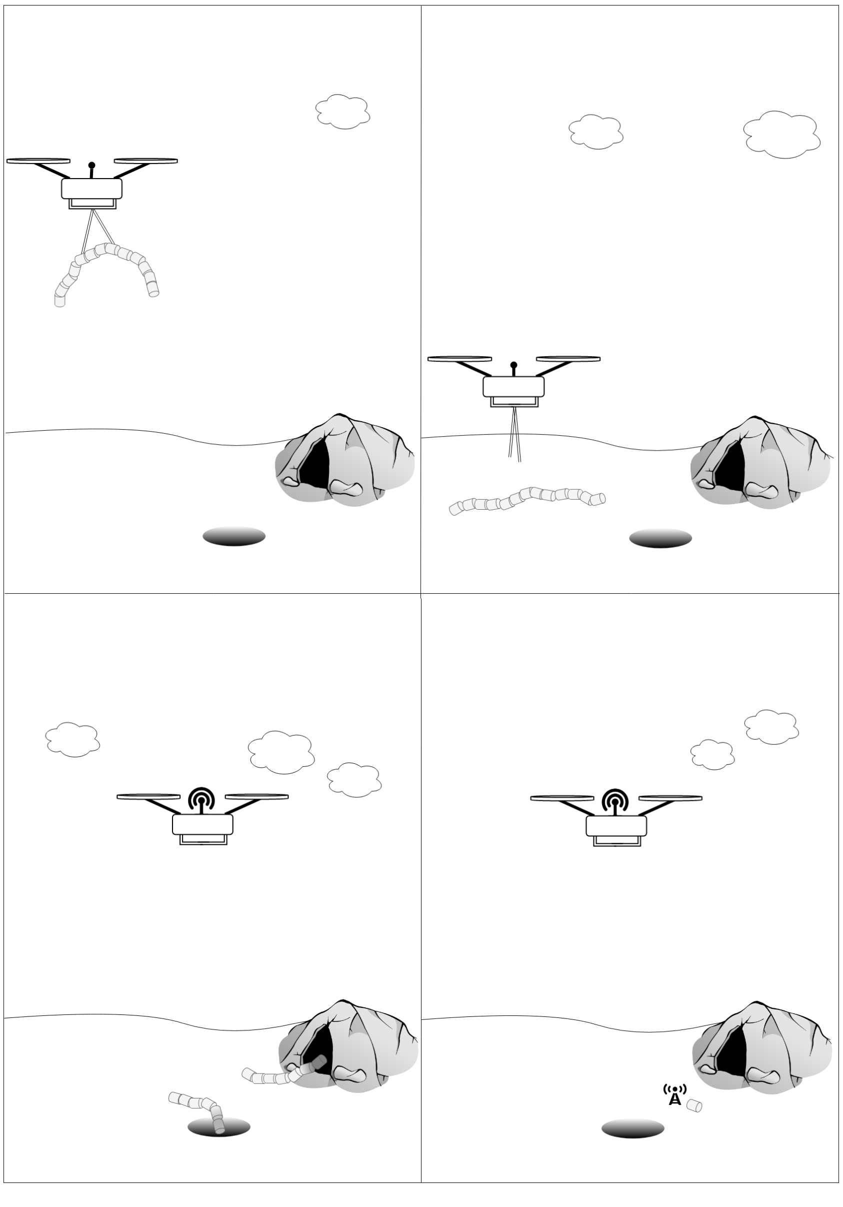


Figure 10.1: A simple illustration of a drone used in collaboration with a snake robot. After deployment, the snake splits up into smaller modules. A simplified portrayal of a transmitting relay is seen in the last image. The relay is meant to be a conceptual illustration of different modules maintaining communication with each other.

10.4 Mesh Networks

Maintaining communication is crucial in certain missions. Establishing reliable communication is not always easy, especially in cluttered areas like urban disaster scenes [8]. These types of areas are candidates for using swarm technology for setting up reliable networks for fast communication. Mesh networks are presented in chapter 2 as a highly effective way of communication. A mesh network can be established by equipping multiple robots with transceivers before sending them out on missions. Because of the robotic snakes' ability to traverse even the most difficult of terrains with ease, make them perfect for these types of missions. They can spread out and cover a large area, creating connections to each other like in figure 10.2. The figure shows red squares as snake units and black lines are bidirectional communication. The goal of a mesh network is for each node to connect to as many other nodes as they can to create the most durable network possible.

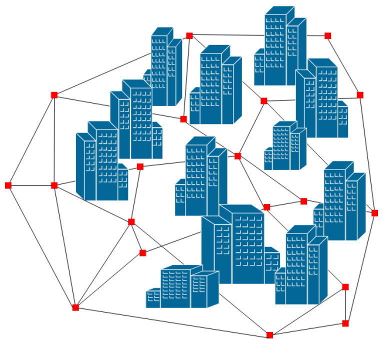


Figure 10.2: Drawing of a non-hierarchically mesh network.

For the robot swarm to know its global positioning in addition to each member's local position estimation relative to each other, GPS receivers can be equipped. Sensor fusion here is a useful technique for fusing internal sensors (e.g. IMU) with external sensors (e.g. GPS). Together with the EKF, an estimation of robot state becomes highly reliable both at the individual and global level [20]. The concept of EKF could be implemented and tested for the current setup, using the OptiTrack camera system as a global observer.

Chapter 11

Conclusion

An improvement of system reliability with the help of optical surveillance was one of the project's main targets for achieving POAL. Modules for gathering and processing visual data are improved and expanded.

With the development and integration of the camera module and the kinematic module, tests regarding optimal marker configurations for position estimation are conducted. Through the experimental results, the kinematic module is changed to fit the most precise calculation algorithm. Sufficient accuracy in estimated joint position implies dependability regarding the movement of the simulated snake.

Two methods of using a camera system for pushpoint extraction are developed, by visual perception alone, and by sensor fusion. Successful testing of both methods demonstrated feasibility and valid system performance. Tests conducted with certain initial conditions showed inadequate system robustness for both methods. The problems concerning the certain initial conditions are addressed and theoretical solutions are proposed as to fix these problems.

The snake in the simulated environment displays authentic sensor values and all necessary data for achieving POAL. Together with an easy to understand visualization makes it ideal for ascertaining accurate performance, and it will work as a tool for future improvements.

Obstacle aided locomotion by sensor fusion is proven plausible in this thesis through experiments. System development and experiments conducted are as of now strictly for laboratory purposes and do not necessarily represent real-life scenarios.

Bibliography

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

[2] Sanfilippo, Stavdahl, and Liljebck, "Snakesim: A ros-based rapid-prototyping framework for perception-driven obstacle-aided locomotion of snake robots," in *Proceedings of the 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Macau, China, 2017.

[3] Liljebck, Stavdahl, Pettersen, and Gravdahl, "Mamba - a waterproof snake robot with tactile sensing," in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, USA, 2014.

[4] "Oldest Robotic Snake," http://www.expo21xx.com/industrial-robots/18224_st3_mobile-robot-research/default.htm, 2018, [Online; accessed 15.03.18].

[5] "OptiTrack," https://www.optitrack.com/products/flex-13/specs.html, 2018, [Online; accessed 15.03.18].

[6] Sanfilippo, Stavdahl, Marafioti, Transeth, and Liljebck, "Virtual functional segmentation of snake robots for perception-driven obstacle-aided locomotion," in *Proceeding of the IEEE Conference on Robotics and Biomimetics (ROBIO)*, Qingdao, China, 2016, pp. 6-10.

[7] "Snake slithering," https://animals.howstuffworks.com/snakes/snake3.htm, 2004, [Online; accessed 12.05.18].

[8] Mobedi and Nejat, "3-d active sensing in time-critical urban search and rescue missions," *IEEE/ASME Transactions on Mechatronics*, vol. 17, 2011.

[9] Kassim, Phee, and Ng, "Locomotion techniques for robotic colonoscopy," *IEEE Engineering in Medicine and Biology Magazine*, vol. 25, pp. 49–56, 2006.

[10] Hitaka and Yokomichi, “Obstacle avoidance of a snake robot in narrow hallway,” in *Proceedings of the 2012 IEEE International Conference on Mechatronics and Automation*, Chengdu, China, 2012.

[11] Shan and Koren, “Obstacle accommodation motion planning,” in *Proceedings of the 1995 IEEE Transactions on Robotics and Automation*, 1995.

[12] Bayraktaroglu and Blazevic, “Understanding snakelike locomotion through a novel push-point approach,” *Journal of Dynamic Systems Measurement and Control*, vol. 127, no. 1, pp. 146–152, 2005.

[13] Liljebck, Pettersen, and Stavdahl, “A snake robot with a contact force measurement system for obstacle-aided locomotion,” in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, USA, 2010.

[14] Fredriksen, *A Sensor calibration and Sensor fusion approach with focus on Tactile force estimation for Perception-Driven Obstacle-Aided snake robot locomotion*. Master’s thesis, NTNU, 2018.

[15] Sanfilippo, Azpiazu, Marafioti, Transeth, Stavdahl, and Liljebck, “Perception-driven obstacle-aided locomotion for snake robots: The state of the art, challenges and possibilities,” in *Proceedings of the 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Phuket, Thailand, 2016.

[16] Oberländer, Klemm, Heppner, Roennau, and Dillmann, “A multi-resolution 3-d environment model for autonomous planetary exploration,” in *Proceedings of the 2014 IEEE International Conference on Automation Science and Engineering (CASE)*, Taipei, Taiwan, 2014.

[17] Ponte, Queenan, Gong, Mertz, Travers, Enner, Hebert, and Choset, “Visual sensing for developing autonomous behavior in snake robots,” in *Proceeding of the International Conference on Robotics and Automation (ICRA) Hong Kong Convention and Exhibition Center (ROBIO)*, Hong Kong, China, 2014.

[18] Karthika, “Wireless mesh network: A survey,” in *Proceedings of the International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, India, 2016.

[19] Kalakrishnan, Buchli, Pastor, and Schaal, “Learning locomotion over rough terrain using terrain templates,” in *Proceeding of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, USA, 2009.

[20] Khatib, Jaradat, and Abdel-Hafez, “Multiple sensor fusion for mobile robot localization and navigation using the extended kalman filter,” in *Proceedings of the 2015 10th International Symposium on Mechatronics and its Applications (ISMA)*, Sharjah, United Arab Emirates, 2015.

[21] “NaturalPoint,” https://www.naturalpoint.com/, 2018, [Online; accessed 15.03.18].

[22] “Motive,” http://optitrack.com/products/motive/tracker/, 2018, [Online; accessed 15.03.18].

[23] “Ros,” http://www.ros.org/about-ros/, [Online; accessed 01.06.18].

[24] “Gazebo,” http://gazebosim.org/tutorials, 2018, [Online; accessed 15.03.18].

[25] Fredriksen, *Visual and Tactile Perception for Obstacle-Aided Locomotion of Snake Robots*. Project thesis, NTNU, 2017.

[26] “Euler to quaternion,” https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19770024290.pdf, 1977, [Online; accessed 16.05.18].

[27] “Atan2,” http://www.cplusplus.com/reference/cmath/atan2/, [Online; accessed 21.05.18].

[28] “Boston Dynamics,” http://fortune.com/2018/02/13/boston-dynamics-robot-spotmini-door/, 2018, [Online; accessed 12.05.18].

Appendix

11.1 Videos from Testing

Videos from experiments are found both in the digital attachments of this thesis and through the links in the following sections.

11.1.1 Videos of POAL

The bullet points list the three attached videos and a link that redirects to a site containing the videos

- POALVisualDistantPushpoints.mp4
- POALVisualReducedPushpointDistance.mp4
- POALTactile.mp4
- `https://1drv.ms/f/s!AuFMzog0Z4MCkwt70V-nTJHV6wd5`

11.1.2 Video of Testing Force Application

The bullet points list the attached force video and a link that redirects to a site containing the video

- ForceTest.mp4
- `https://1drv.ms/f/s!AuFMzog0Z4MCkxAdfXWgwB8d61x_`

11.1.3 Literature Review Papers

Several papers used in the literary research are found through the following link

- `https://1drv.ms/f/s!AuFMzog0Z4MCk3igL3Ud_IVXwrxi`