TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Andreas Nagel 192207IAPM

Blind Mapping and Localisation for Small-Scale Mining Robots

Master's thesis

Supervisor: Asko Ristolainen
PhD
Co-supervisors: Laura Piho
PhD
Roza Gkliva
MSc

Tallinn 2021

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Andreas Nagel 192207IAPM

Pime kaardistamine ja lokaliseerimine väikesemõõdulistele kaevandusrobotitele

Magistritöö

Juhendaja: Asko Ristolainen
PhD
Kaasjuhendajad: Laura Piho
PhD
Roza Gkliva
MSc

Tallinn 2021

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Andreas Nagel

17.05.2021

Abstract

An autonomous robot is required to have the means to map the environment and localise itself within the environment for successful operation. The solution to this challenge is dependent on the sensors available to the robot and the environment. Common state-of-the-art approaches solve the challenge by using visual sensors in a SLAM framework.

In this thesis, the solutions to the online mapping and localisation problem in low-visibility and multi-phase conditions are investigated on utilizing a passive tactile whisker sensor grid. A proof-of-concept is provided on the basis of a simulation by showing the accuracy of a SLAM algorithm compared to odometry. The approach is shown to be effective in simulation over varying scenarios of movement, with the proposed SLAM algorithm achieving higher pose estimation accuracy over pure odometry.

The results of this thesis provide a basis for real-world experiments and further improvements on the developed algorithms. Furthermore, the results improve the possibility to adopt an unconventional sensor modality for autonomous small-scale mining robots.

This thesis is written in English and is 44 pages long, including 5 chapters and 31 figures.

Annotatsioon

Pime kaardistamine ja lokaliseerimine väikesemõõdulistele kaevandusrobotitele

Keskkonnas liikumiseks vajavad autonoomsed robotid vahendeid ümbritseva kaardistamiseks ja enda lokaliseerimiseks. Kaardistamise ja lokaliseerimise meetodi valik sõltub töökeskkonnast ning robotil kasutatavatest sensoritest. Kaasaegsetes lahendustes kasutakse erinevaid visuaalseid sensoreid samaaegse lokaliseerimise ja kaardistamise (SLAM) raamistikes.

Käesolevas töös kasutatakse passiivset taktiilset tehisvurrudel baseeruvat sensormaatriksit, et uurida lahendusi reaalajas kaardistamise ja lokaliseerimise probleemile halvas nähtavuses ning mitmefaasilistes tingimustes. SLAMi algoritmi kontseptsiooni testimine ja valideerimine teostatakse simuleeritud keskkonnas, võrreldes SLAMi algoritmi täpsust roboti odomeetriaga. Antud töös näidatakse, et valitud lähenemine on efektiivne erinevate liikumis-stsenaariumite juures, kusjuures loodud lahenduse asukohahinnang on täpsem odomeetria abil leitud asukohahinnangust.

Käesolev lõputöö loob aluse edasisteks reaalseteks eksperimentideks füüsilise sensormaatriksiga ning edasisteks täiendusteks väljatöötatud algoritmile. Lisaks sellele pakuvad tulemused võimalusi uut tüüpi sensori kasutuselevõtuks autonoomsetele väikesemõõdulistele kaevandusrobotitele.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 44 leheküljel, 5 peatükki ja 31 joonist.

List of abbreviations and terms

CAD	Computer-aided design
DOF	Degree of Freedom
GB	Gigabyte
GICP	Generalized Iterative Closest Point
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
GTSAM	Georgia Tech Smoothing and Mapping
GUI	Graphical User Interface
IMU	Inertial Measurement Unit
Lidar	Light Detection and Ranging
LOAM	Lidar Odometry and Mapping
NDT	Normal Distributions Transform
PCL	Point Cloud Library
RAM	Random access memory
RGB-D camera	Red Green Blue – Depth camera – a colour camera with distance measurements incorporated with the picture.
ROS	Robot Operating System
RTAB-Map	Real-Time Appearance Based Mapping
SDF	Simulation Description Format
SLAM	Simultaneous Localisation and Mapping
SoC	System on a Chip
URDF	Universal Robot Description Format
USB	Universal Serial Bus
XML	Extensible Markup Language

Table of contents

List of figures

1 Introduction

Online localisation and online mapping are fundamental challenges in autonomous mobile robotics. For the operation of most autonomous mobile robots, the robot is required to know its location with respect to the surroundings. This, however, is an interconnected problem, due to its nature. To reliably map the surroundings, the robot is required to have an accurate location of itself, while for an accurate localisation with respect to environment, it is required to have a well-defined map of the surroundings. The state-of-the-art approaches to tackle this problem often take the form of an iterative simultaneous localisation and mapping (SLAM) algorithm, where localisation operation is interleaved with the mapping operation. This approach allows the robot to have better location estimation and increase the mapping accuracy with each new iteration.

The environment for mobile robot operation and choice of sensors also plays an important role in the effectiveness the SLAM algorithm chosen. Often, the measurements from multiple sensors are combined to increase the accuracy. These sensors can be categorised as exteroceptive or proprioceptive, based on the states they are measuring. Exteroceptive sensors gather information about the environment, measuring values external to the robot, while proprioceptive sensors measure values internal to the robot.

Proprioceptive sensors, such as an inertial measurement unit (IMU), wheel encoders, or joint position sensors enable to estimate the robot's location in a process referred to as "dead reckoning". These sensors are commonly used for localisation but not mapping, because by definition, they measure values internal to the robot, not the environment. This method of localisation is prone to drift from the correct location of the robot as their results are integrated and therefore the error cumulatively increases over time. There exist also other sources of errors, for example, the joint position sensors rely on the interaction between the robot and environment to be perfect, with no losses. However, this is almost never the case, thus, the error for these sensors is unbounded and accumulates to infinity.

Exteroceptive sensors, such as camera, lidar, and GPS, are used to collect information about the environment and use this information to localize the robot. These sensors provide means to accurately map and localize a robot simultaneously without accruing as much drift as with proprioceptive sensors, and a way to correct for it online. First sensor reading can create a base for a map of the robot's surroundings and each new localisation

can determine the current distance of the robot from objects on the map, simultaneously updating the map.

While it is possible to get a reasonable estimation of robot's global location with a GPS module in an open environment, it can be less reliable in indoor conditions, or downright impossible in an underground or underwater environment. Similarly, using a camera during daylight in outdoor conditions can lead to good results, while in low visibility conditions, frequently changing visual conditions, or visually self-similar environments, it is more difficult to get good results from a vision-based SLAM algorithm. Lidar based SLAM also suffers in poor visibility conditions, such as in foliage, smoke, dust, or otherwise multi-phase environments. In constrained environments tactile sensor-based SLAM has been used, which could offer good results in low visibility and possibly multi-phase environment conditions.

Three very commonly used sensors for SLAM are cameras, lidars and RGB-D cameras, the first of which provides visual RGB colour information, the second providing depth information of unoccupied space from the sensor, and the last being a hybrid of the two, providing both visual colour information and depth information of objects in recorded video.

ROBOMINERS is a Horizon 2020 project that is developing a bio-inspired robot-miner for small and difficult to access mineral deposits [1] (www.robominers.eu). The aim of the project is to create a prototype robot capable of mining underground deposits both above and under water. Due to impaired visibility in this environment, it is envisioned to have artificial whisker sensors for situational awareness, and navigational purposes to mine and retrieve mineral deposits. To navigate these underground environments the robot needs to have a sense of its surroundings and its location relative to them. Therefore, there exists a need for a mapping and localisation framework that uses these whisker sensors in addition to other sensors perception options that the robot might have, such as temperature, pressure sensor, IMU, and wheel encoder-based odometry.

This thesis will investigate the problem of localisation and mapping of a robot in a challenging environment using non-visual sensors. The aim of this work is to provide a proof of concept of a mapping and localisation framework, that works in conditions similar to a deep underground mine, which is possibly partially submerged. In looser

terms this is regarded to as “blind mapping”. To achieve this goal the robot will use bioinspired whiskers developed during the project [1].

The contributions of this thesis are the following:

- Development of a whisker sensor grid model for use in a simulated environment.
- Development of a framework for testing the SLAM algorithm in the simulation.
- Development of a SLAM algorithm using the whisker sensor array and evaluation in a simulation.

2 Background

This chapter offers background information on the robot prototype and the sensor used for SLAM purposes. Additionally, a literature review of state-of-the-art SLAM frameworks is presented, with focus on those relevant to thesis. To conclude the overview, SLAM approaches in either underground environments, or using whisker sensors are presented.

2.1 ROBOMINERS platform

The ROBOMINERS autonomous platform (Figure 1) is being developed during the Horizon 2020 project. The project is ongoing as of the writing of this thesis. The aim is to develop a prototype robot with capabilities to mine minerals autonomously deep underground in locations hard to reach with conventional mining technology.

For locomotion, the platform uses Archimedes screws, which give it capabilities to move omnidirectionally on rough ground, gravel, sand, above water, underwater, and on slurry. The test platform's control system is distributed across multiple Olimex A20 SoC modules, which have a multicore ARM processor and 1 GB of RAM. This sets some constraints for the SLAM algorithm computationally.

Figure 1. Robominer test platform with drive screw actuators and whisker sensor array attached. Left figure shows a small-scale prototype. Right figure shows a CAD assembly of the prototype.

The operating system running on the final prototype will be Ubuntu 20.04 running ROS 2 Foxy Fitzroy. However, for running a simulation of the ROBOMINER whisker sensor, ROS 1 Noetic is used as not all of the functionality of ROS 1 has been ported to ROS 2.

The sensors on the platform aimed to be used for SLAM purposes include the whisker array developed by TalTech Centre for Biorobotics, an IMU, and encoders from the screw actuators for odometry estimation. Additional sensors to consider within the scope of the Horizon 2020 ROBOMINERS project, which will not fit into the timeframe of this thesis, include an x-ray fluorescence scanner, temperature and pressure sensors, wheel motion sensor, or material conductivity sensors, to augment environmental data collection.

2.1.1 Whisker sensors

The whisker sensor (Figure 2) consists of a 3D Hall sensor and a magnet mounted on the base of the "whisker". It is inspired by the hydromast sensor [2]. The whisker is fixed to a silicone membrane, which acts as a spring element, returning the whisker to its resting position when no outside forces are acting on it. The 3D Hall sensor reports a magnetic field vector in its location. Knowing the resting position of the whisker, it is possible to find the position of the whisker, when it is bent to some other position. The whisker array reports inclination and azimuth of each whisker in the grid (Figure 3). For a fixed length whisker, it is then possible to calculate the endpoint of whisker in relation to the whisker base location and use this as an estimate of a contact point with another object. For the purposes of this thesis, the inclination angle is measured from the z-axis of the whisker, while the azimuth is measured from the x-axis.

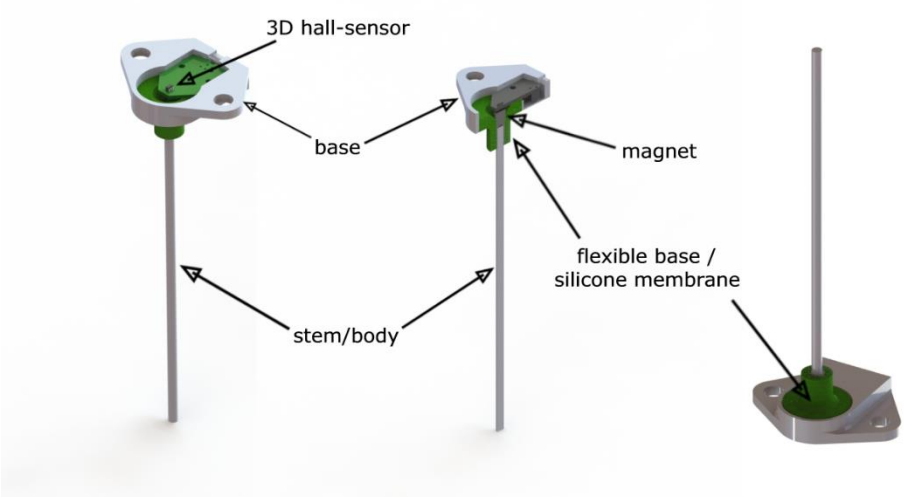

Figure 2. The whisker sensor from various views.

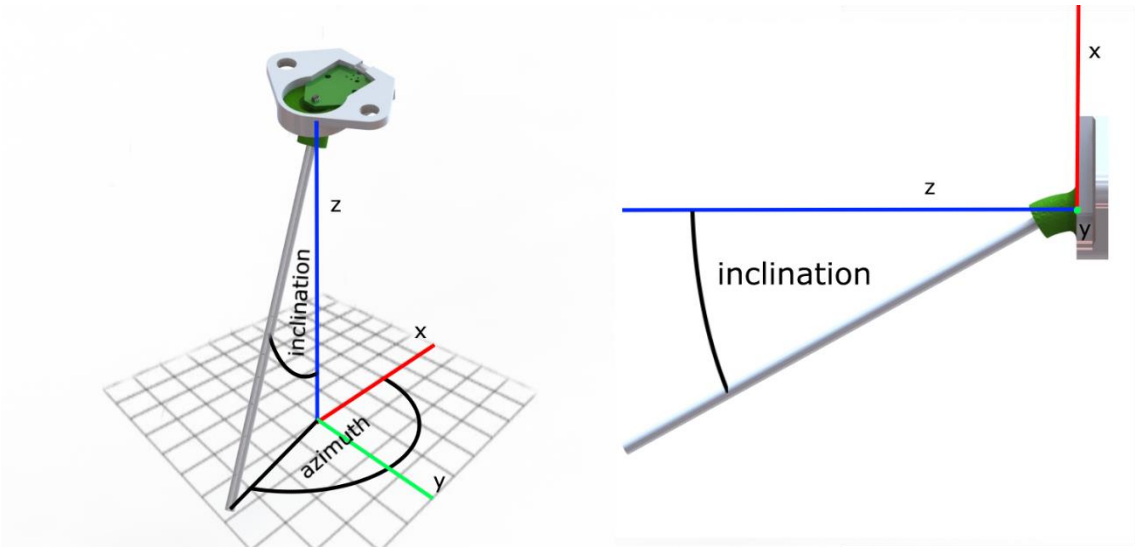

Figure 3. Azimuth and inclination of the whisker sensor.

When whiskers aligned in a grid come in contact with an object, it is possible then to map a part of a surface which is in contact with the whiskers on the grid, by finding the endpoints of the whiskers.

The whisker length can be varied, as can be the whisker stiffness by choosing a whisker made of a different material. The whisker sensor grid alignment and individual sensor positions can also be varied. Currently the sensor grid is set up as 8-by-8 flat grid of sensors with distances between individual sensors being 39 mm and 64 mm in x-, and y-coordinate respectively in robot frame (Figure 9). In the current configuration, every second row of sensors in the x-axis is shifted by 32 mm in the y-axis, forming a chess-board-like pattern (Figure 9), to maximize coverage.

2.2 **SLAM**

SLAM is an open research question in various different environments with several different sensors. It consists of two subproblems – localisation and mapping – which each also have their own subproblems. A SLAM approach can be classified as one of two approaches – filtering or smoothing[3], with the smoothing approaches sometimes referred to as bundle adjustment type optimisation[4]. Filtering approaches, such as Extended Kalman Filter SLAM and particle filter SLAM, model the problem as an on-line state estimation problem, estimating only the current state of the robot and refining the map, while smoothing approaches, such as different graph-based SLAM algorithms [5], estimate the full trajectory of the robot according to the full set of measurements.

Both approaches build the map of the environment in a form of a graph, but the structure of the graph differs [4].

Filtering approaches marginalise the previous robot positions and build a graph which links the features extracted from sensor measurements [4]. This approach allows to keep the graph relatively compact, since the size of the graph depends only on features observed. Smoothing approaches on the other hand build a graph which link the previous robot poses with features extracted from sensor measurements. To keep the size of the graph under control, only some of the past poses with their related measurements are added to the graph, the rest are discarded. These chosen poses with their respective measurements are referred to as keyframes.

A SLAM approach can usually be divided into two sections – SLAM backend and SLAM frontend [3]. SLAM frontend focuses on interpreting the sensor data to create constraints between measurements, while SLAM backend focuses on computing the best map according to sensor data and constraints between measurements.

A number of SLAM frameworks for different use cases have been developed and tested by researchers in recent years. A selection of these considered for use during thesis is described in the next section.

2.2.1 Popular frameworks

RTAB-Map (Real-Time Appearance Based Mapping) is a real-time graph-SLAM approach based on incremental appearance-based loop closure detector [6]. It is capable of integrating RGB-D camera, stereo camera, and lidar measurements. In addition, it supports odometry readings from outside the framework, meaning it can integrate IMU, wheel odometry or GNSS readings. In minimal configuration it can work with only a stereo camera or an RGB-D camera. It can also work with only a lidar sensor, but in that configuration, it forgoes the ability to perform loop closures, which it performs with visual data input.

RTAB-Map uses a visual bag-of-words approach for finding loop closures, which works well for even large loop closures where location estimation has drifted greatly from ground truth. It can return its current map in OctoMap, point cloud, 2D occupancy grid

map, or in its map graph format. RTAB-Map works with a wide range of sensors, is modular and well documented, with source code available on GitHub.

Lidar Odometry and Mapping (LOAM) is a real-time SLAM approach based on lidar readings proposed by Zhang and Singh [7]. It is an online approach, that constructs a map of the robot's surroundings by registering the point clouds measured by the lidar sensor. It performs two different frequency scan matchings to align the measured point clouds and localise the sensor in the environment. The faster scan matching forgoes some accuracy in order to keep track of the robot's velocity and make sure the localisation works in real time. The slower scan matching is more accurate and provides a more fine-tuned scan matching to update the map representation and robot's location on the map. The algorithm produced high accuracy odometry and point-cloud maps according to the above-mentioned paper.

The approach by Zhang and Singh in its original form only provides odometry from consequent lidar measurements, which means that it is susceptible to small drift in location accuracy, which will accumulate over time. It does not perform loop closure to correct the map representation and is therefore not accurate in case of large loops in the environment, where the robot location estimation has drifted too much to match new incoming lidar scans of the same location with previously performed scans.

SegMatch is a segment-based place recognition algorithm for 3D point clouds [8]. It is designed to perform real-time loop closures for a graph-SLAM algorithm using point cloud data as an input. The most common sensors generating point clouds are lidars, therefore this method is meant to correct for large errors in lidar equipped robot's localisation estimates, when the robot revisits an already mapped location.

The algorithm first segments the incoming point cloud, by removing the ground points and then clustering nearby points, to form a segment [8]. Then, features consisting of eigenvalue-based features and ensemble of shape histograms are extracted for each segment. For the third step a classifier is trained to match segments based on the previously extracted features. The original approach uses a random forest classifier, which assigns each segment a score of being a match with a previously scanned segment. Finally, the candidate matches found by random forest classifier are geometrically

verified based on the location of segment centroids. If the location of segments is similar enough, the match is considered successful and robot pose can be updated.

OctoMap is a 3D occupancy grid mapping framework, which can be used in conjunction with other SLAM algorithms [9]. It stores a voxel grid map in a tree structure, making it efficient to use for large scale 3D maps. Each node in OctoMap represents a volume of space. Each partially occupied node in the map has eight child nodes, each representing one eighth of the parent node's volume. This allows OctoMap to retain high resolution while also being memory efficient compared to saving the full grid, which is the preferred option for 2D mapping. OctoMap can be used in conjunction with a path planning or localisation algorithm. It can also incorporate other types of data in addition to occupancy value, such as for example RGB color data, or other custom user-defined data.

2.2.2 SLAM in underground environments

The previous section provided a literature review of generic localisation and mapping approaches. To get a better sense of the environment that the ROBOMINERS robot will have to perform in, it is useful to look at localisation and mapping approaches used in a similar underground environment.

One solution to underground localisation problem is provided by Jacobson *et al* [10]. They present a multisensor localisation framework that provides suitable accuracy for performing autonomous operations. Their approach uses camera, laser sensor and odometry data which is fed into a particle filter architecture to produce an accurate position estimate within underground mines. They also note, that producing maps within underground environments is complicated due to environmental conditions and they used the help of a human operator to manually ground the location of the vehicle at identifiable landmarks. This meant that the mapping process is performed once with the help of an operator and the produced map is suitable for localisation. During the mapping phase they used GTSAM [5], which is a computationally efficient implementation of graph-based SLAM algorithm [3]. Their work produced a localisation approach with mean localisation error of 1.32 m in their New South Wales dataset, which consisted of 32.8 km of vehicle travel data, which is suitable for underground autonomous robotics.

2.2.3 SLAM using whisker sensors

Another useful viewpoint to have is to have a look at localisation and mapping approaches taken with similar types of sensors.

An early approach used active whisker sensor array to solve 3 DOF simultaneous mapping and localisation problem [11]. It consisted of a mobile platform with 18 whiskers mounted on a custom-built head and neck. Their approach used a particle filter to track the robot location and recorded whisker sensor measurements in an occupancy grid map and consistently achieved better than raw odometry results in localisation estimate while performing simultaneous localisation and mapping.

Later the Whisker-RatSLAM [12] uses a similar array of active whiskers in an attempt to solve 6 DOF mapping and localisation problem, as well as object identification. Whisker-RatSLAM uses various bio-inspired active whisking strategies to generate contact point information from its whisker sensors. The whisker sensors record whisker rotation angle, and deflection forces recorded at its shaft in two perpendicular frames orthogonal to the whisker shaft, which are used to calculate an estimation for contact points for each whisker. From contact points two general features are extracted – the point feature histogram and slope distribution array – which are later used to perform localisation of the sensor in environment. The approach reports mostly correct localisation results with some incorrect localisations due to the test objects used being highly symmetrical.

3 Methodology

This chapter describes the methods used to achieve and evaluate the results presented in chapter 4. First, a high-level overview of the ROS-based simulation framework and SLAM algorithm approach is presented, which is followed by the description of validation process. Additionally, necessary implementation details are presented in section 3.4, to allow replication of the results.

The robot on which the simultaneous mapping and localisation framework must work is still being developed. The first sensor array prototype was completed during the writing of this thesis. Due to this, the SLAM framework using the whisker sensor array was developed and evaluated mostly in simulation.

3.1 Simulation framework

There are a few different simulators which are straight-forward to interface with ROS [13] such as V-REP [14], Gazebo [15], and WeBots [16]. The simulator chosen for this thesis was Gazebo, since the author has experience with this simulator.

Gazebo is an open-source simulator supporting a GUI and multiple physics engines. Gazebo is tightly integrated with ROS, enabling to control the robot and simulate different sensors.

The original intent was simulating the whisker array attached to a mobile base powered by the Archimedean screws with the same scale and size as the prototype built in the TalTech Centre for Biorobotics (Figure 1). Unfortunately, the simulator could not realistically simulate movement powered by the Archimedean screws due to very limited contract surface they exhibit. Therefore, only the whisker sensor array was used in the simulation (Figure 4). This change also allowed to focus directly on simulating the whisker sensor and helped increase simulation speed due to lower number of calculations required.

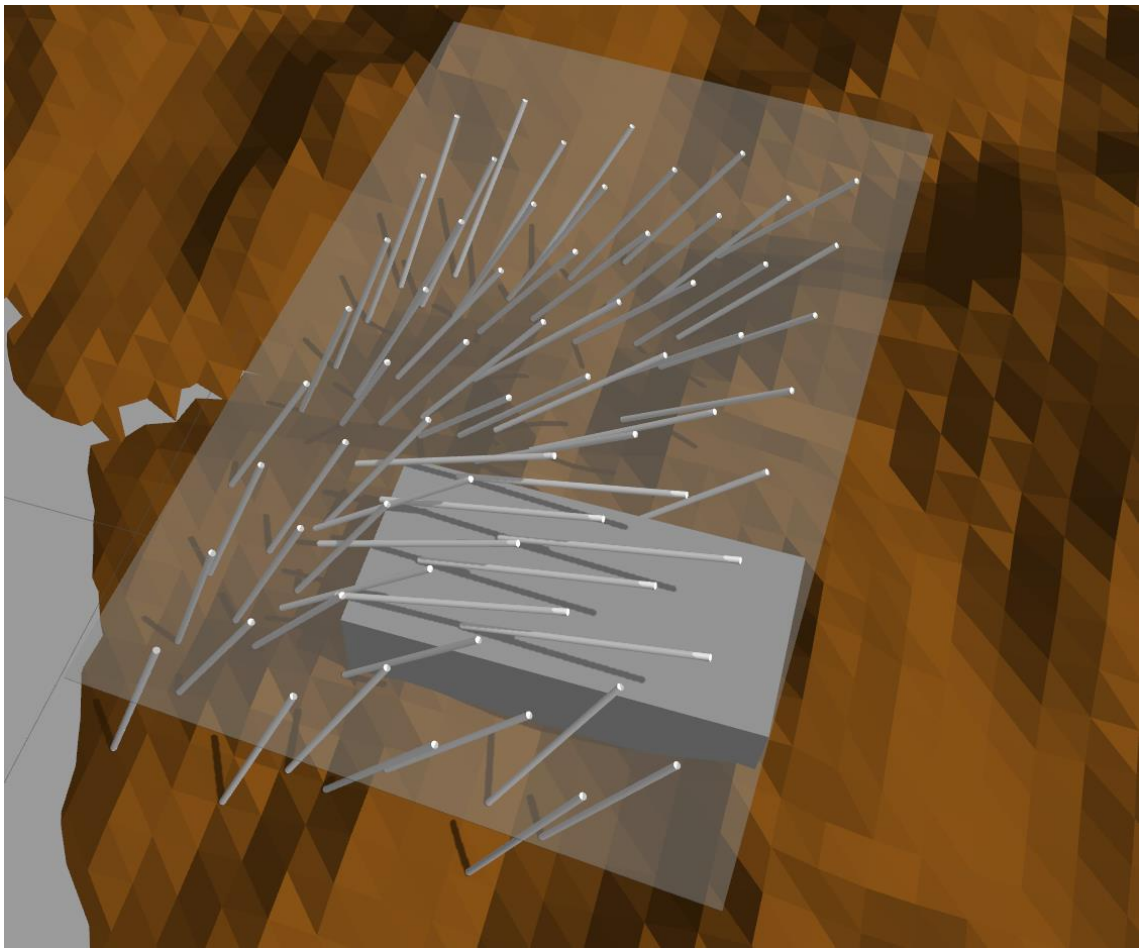

Figure 4. The whisker sensor in simulator.

In the simulation environment, the whisker sensor array is mounted on a grid and kept level at the height of the whiskers. The grid can move holonomically, similar to the movement of the screw driven robot platform. It is then moved over an uneven terrain model, populated with some geometric shapes such as a cuboid in different orientations, a sphere, and a cylinder (Figure 5), by means of a waypoint controller. The waypoint controller reads in an array of waypoints and guides the robot to them in a predefined sequence.

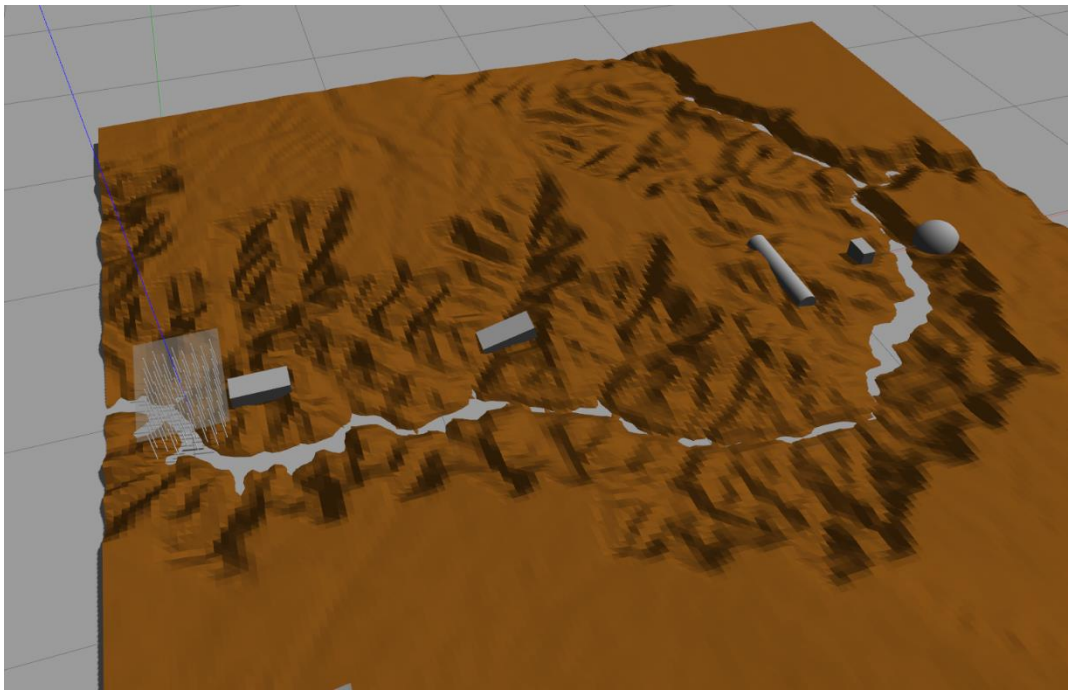

Figure 5. The uneven terrain with geometric shapes and the whisker sensor in the Gazebo simulator.

During the movement, the simulated whisker sensor array publishes joint angles for each of the whisker sensors (Figure 6). These joint angles are transformed into inclination and azimuth angles, to match the data format of the physical sensor. From the published angles, a point cloud of contact points is generated assuming the whisker tip as a contact point with the environment.

An odometry estimation is calculated from the simulation ground truth by calculating the sensor grid's velocity and adding noise to it. This was done, to simulate a noisy biased odometry reading, that the vehicle would have in real conditions. The odometry and sensor readings are fed to the SLAM algorithm, which then maps the sensed area and attempts to correct sensor pose estimation (Figure 6).

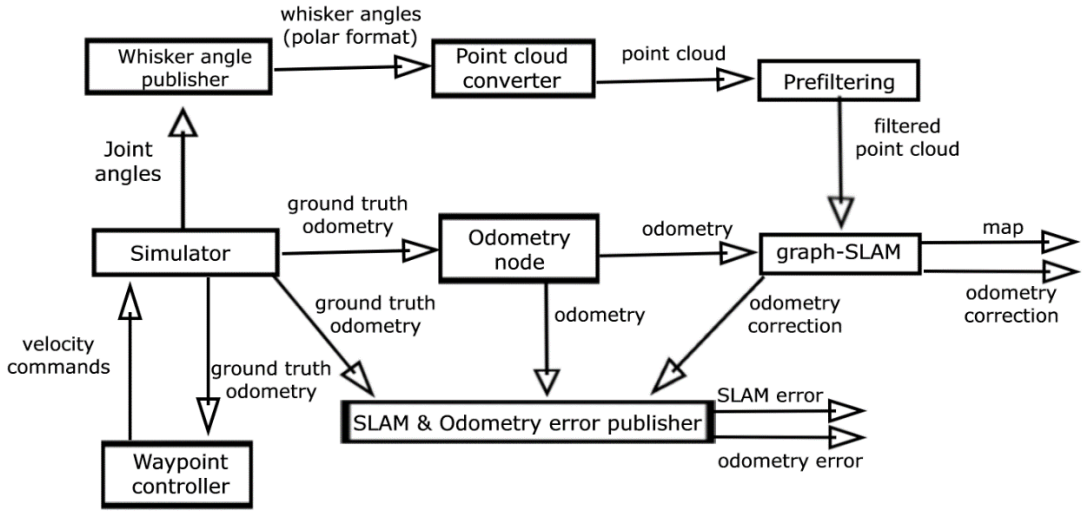

Figure 6. Flowchart of information flow between vehicle framework components.

For evaluation purposes, an error publishing node was created, which measures the pose error in 6D (x, y, z, roll, pitch, yaw) between ground truth and pure odometry, and between ground truth and SLAM corrected odometry (Figure 6). This allows to evaluate the performance of the SLAM algorithm online as well as in post processing. Since pure odometry is capable of estimating only the x, y, and yaw components of the pose error, the evaluation was based on these three components of the pose error.

3.2 **SLAM – description of approach**

Due to the whiskers being passive on our sensor it can be argued that the approach described by [12] is unlikely to work. Salman *et al* relied on the dense point cloud and accurate prediction of whisker behaviour to generate a view of environmental features, while the ROBOMINERS whisker array is more sparse, and unable to generate similarly dense and accurate point clouds. Moreover, the Salman *et al* did not actually discuss simultaneous localisation and mapping but focused on localisation and object recognition based on an existing map in simulation.

The artificial whisker array also does not report any data similar to colour or brightness, that is used in camera-based SLAM frameworks. Due to these limitations, camera-based frameworks are also unlikely to work with the sensor array. It was decided that the passive whisker sensor array used for the ROBOMINERS project most closely resembles a lidar with very limited range and field of view, among the commonly used sensors for SLAM. The SLAM framework was then also modelled after those used with lidar sensors.

The SLAM algorithm was based on an algorithm described by [17]. The source code from GitHub was used and adapted for use with the whisker array sensor. Although the described SLAM framework was used offline in their paper, it was used online during this thesis with the whisker sensor. It can be used online due to the considerably smaller point cloud input from the whisker sensor, compared to the lidar that was used during the paper, and due to the limited size of the simulated environment.

3.3 **SLAM evaluation**

To evaluate SLAM algorithm's performance, it was decided to move the sensor array in the environment according to 5 different scenarios, which consist of movements between the waypoints. Each scenario was performed once, except for scenario 3, which was performed 10 times, to investigate the repeatability of the experiments. The waypoints can be seen on Figure 7, sensor starting position is denoted with S. The sensor starts with its x and y-axis aligned with the x and y-axis of the map. All rotations mentioned in this section are performed around robot's z-axis.

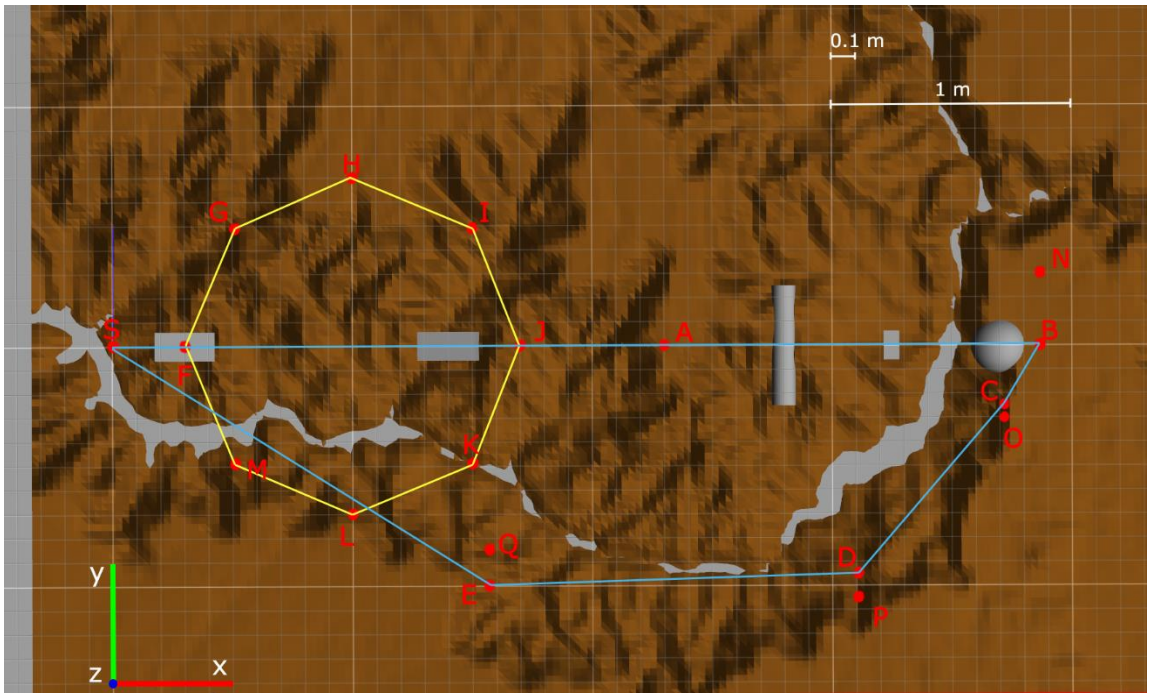

Figure 7. The waypoints on simulation map. The small and large loop denoted with yellow and teal lines respectively. Grid scale visible top right. Axes shown bottom left: red – x-axis, green – y-axis, blue – z-axis pointing upwards.

Let small loop (visible in yellow on Figure 7) be a movement starting in point F with robot's x-axis coinciding with y-axis of the map and moving from point F to G to H to I to J to K to L to M and back to F, while rotating 45 degrees clockwise between each transition between points.

Let large loop (visible in teal on Figure 7) be a movement starting in point S with robot's x-axis coinciding with map x-axis and moving from S to B, from B to C while rotating 90 degrees counterclockwise, from C to D while rotating 180 degrees counterclockwise, and from D to E to S.

In each of the scenarios the robot starts its movement from waypoint S. The descriptions of movements performed in each scenario are following:

1. In scenario 1 the robot moves from S to A to S to B. In B it performs a full counterclockwise rotation and then moves back to S. Back in S it performs a 180-degree rotation and moves to B and then back to S. This concludes the first scenario.
2. In the second scenario the robot moves from S to F, where it turns 90 degrees counterclockwise. Then it performs the small loop twice in sequence after which it returns to starting point S while turning 90 degrees counterclockwise.
3. Scenario 3 consists of scenario 2, followed by scenario 1.
4. Scenario 4 starts by following scenario 2. By the end of scenario 2, the sensor is in waypoint S rotated by 180 degrees from its original orientation. Then the sensor is guided to B and back to S, where it rotates 180 degrees returning to its starting pose. It then performs a large loop, after which it rotates 180 degrees and goes from point S to B.
5. Scenario 5 starts with a variation of the large loop, where points B, C, D, and E have been replaced with points N, O, P, and Q respectively. After the variation it performs the original large loop twice in succession. After these the sensor moves from S to B to S to C, where it rotates 90 degrees counterclockwise. Then it performs the small loop twice after which it moves to S while turning 90 degrees counterclockwise and then moves from S to A and back to S.

The scenarios are created to be repetitive on purpose, visiting known locations can help the SLAM algorithm correct its pose estimation via loop closures. Visiting previously

mapped locations while odometry pose estimation error has not grown large, gives a higher chance of correct relocalisation.

The mapped environment (Figure 8) consists of uneven terrain and five simple geometric shapes, collectively referred to as artificial objects. Starting from left on the map, the objects individually are referred to as: brick, slide, pipe, stone, and hemisphere.

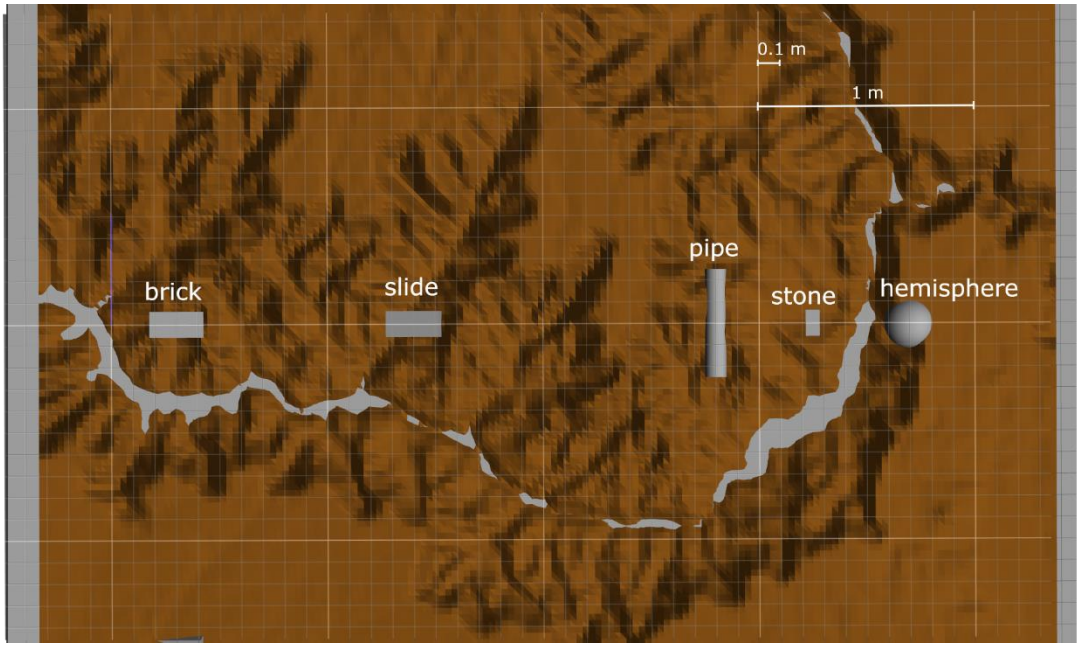

Figure 8. Top-Down orthographic view of map in simulated environment. Large grid squares have 1 metre side length, small grid squares have a side length of 0,1 meters.

3.4 Implementation details

3.4.1 Whisker sensor model in simulation

A description of the whisker sensor used in simulation and with ROS was written in Uniform Robot Description Format, using xacro. URDF is an XML format for representing a robot model and xacro is an XML macro language, which allows to write more compact robot descriptions. The xacro file is used to generate an URDF file, which is used by various ROS plugins and tools. The URDF file itself is used to generate an SDF file by Gazebo simulator, which is then used to generate the robot in simulation.

The URDF description of the robot consists of links and joints and their physical parameters which are used in simulation. The links describe rigid parts of the body, while joints describe the connections between links. The description is written in a tree

structure, such that each link except for the root link of the robot, has one parent link. Links and joints both have a selection of parameters to describe the bodies and their behaviour in the simulation. Each link can have inertial, visual and collision properties.

During the starting phase of the thesis the whisker sensor model was created for use with the Gazebo simulator. The whisker grid consists of a grid base, which is a thin flat surface, similar to a thin steel sheet and whiskers connected to it via joints. For purposes of visualisation, the grid base is made transparent in the simulation. The inertial tensor values of the 3D Hall sensor, magnet and whisker base were exported from the Solidworks model of the whisker sensor and added to the simulation.

The simulation of collisions between the flexible whisker base and more rigid whisker body are not simulated efficiently nor accurately in Gazebo, as it is meant to simulate rigid body dynamics. Therefore, the visual and collision properties for the sensor and flexible silicone membrane were removed from the robot description to reduce the number of objects needed to simulate. Instead, the flexibility was simulated by setting spring stiffness and viscous damping coefficients to the joints connecting whiskers to the grid.

Each whisker in the array is tied to a grid via two revolute joints, first of which allows the whisker to rotate with respect to the x-axis, and the second with respect to the y-axis in robot frame (Figure 9). These revolute joints both have a range of motion between $[-\pi; \pi]$. The result of this configuration is a ball-joint, which gives each whisker a range of motion that covers a hemisphere. The whisker stems themselves are modelled after a steel rod with the length of 150 mm. This coincides with the length used for the prototype-in-construction, but the whisker stems for the first prototype were made of plastic to minimize the risk of damage to the sensors.

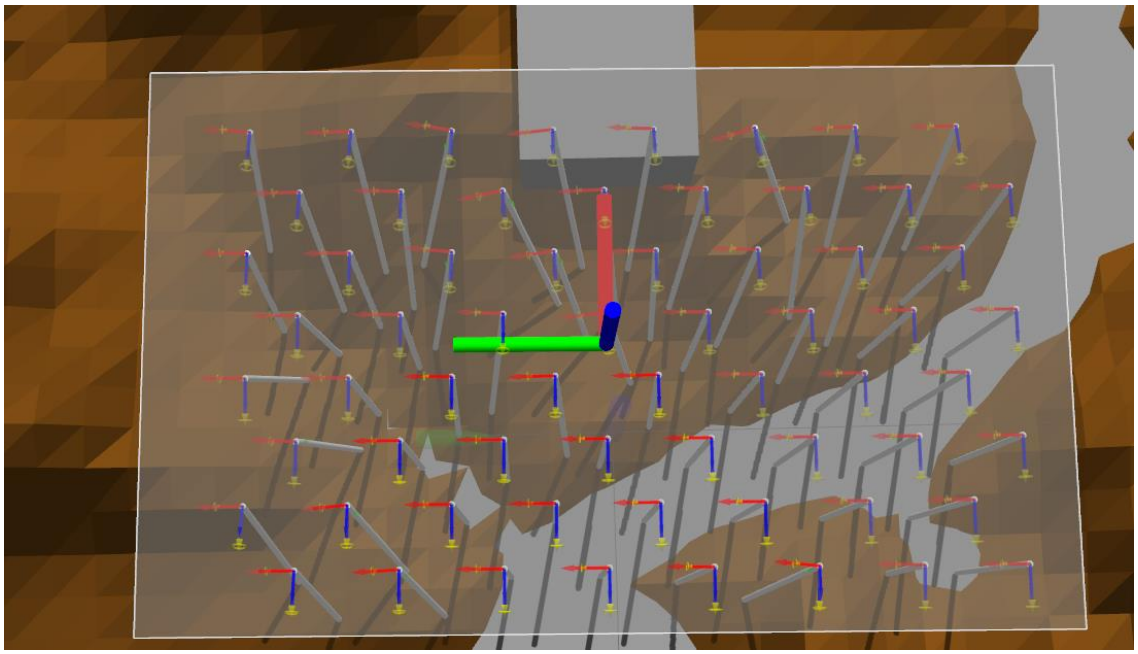

Figure 9. Whisker sensor robot frame and joints. The robot frame is denoted by the large central axis. Whisker joints axes denoted by each of the smaller set of axes. Direction of whisker rotation denoted by yellow circular arrowheads. Axis colours: red – x, green – y, blue – z.

To accurately simulate the stiffness of the joint, the spring stiffness constant of a prototype whisker was calculated and added to the whisker description in the simulation (Figure 11, Figure 12). Also, a rough estimate of viscous damping coefficient was made based on the whisker resonant frequency, and this value too was added to the simulation. Both these processes are described in section 3.4.2.

This resulted in a simulated sensor array behaving visually similar to the prototype that was later built. Then a joint state controller [18] was configured to publish the joint angles for both of the revolute joints during the simulation. A ROS node was programmed to read the joint angles and turn them into azimuth and inclination values, similar to the output of the prototype being built. Another ROS node was programmed which turned azimuth, inclination, and sensor location relative to the sensor base, into a point cloud measurement relative to the sensor base, as it is common for lidar sensors. A flowchart covering this process is visible in Figure 6.

The joint angles initially reported for the whisker sensors by the joint state controller were inaccurate, especially for low angles (Figure 10). The angle readings fluctuated wildly between the positive and the negative value of the same magnitude. Further investigation indicated that the simulator itself was at fault, or the physics engine behind it, because it

was the simulator, that reported the fluctuating angle values to the joint state controller. To address the issue, the joint angles for the whisker sensor were calculated from the global poses of the whisker joint and whisker body, as reported by the simulator. The angles calculated in this manner were as accurate and did not exhibit the fluctuating behaviour. To affect every whisker sensor joint efficiently, the changes were made directly to the hardware driver plugin [19], which is connecting the joint state controller to the simulator, therefore it is necessary to compile this package from source with the added change.

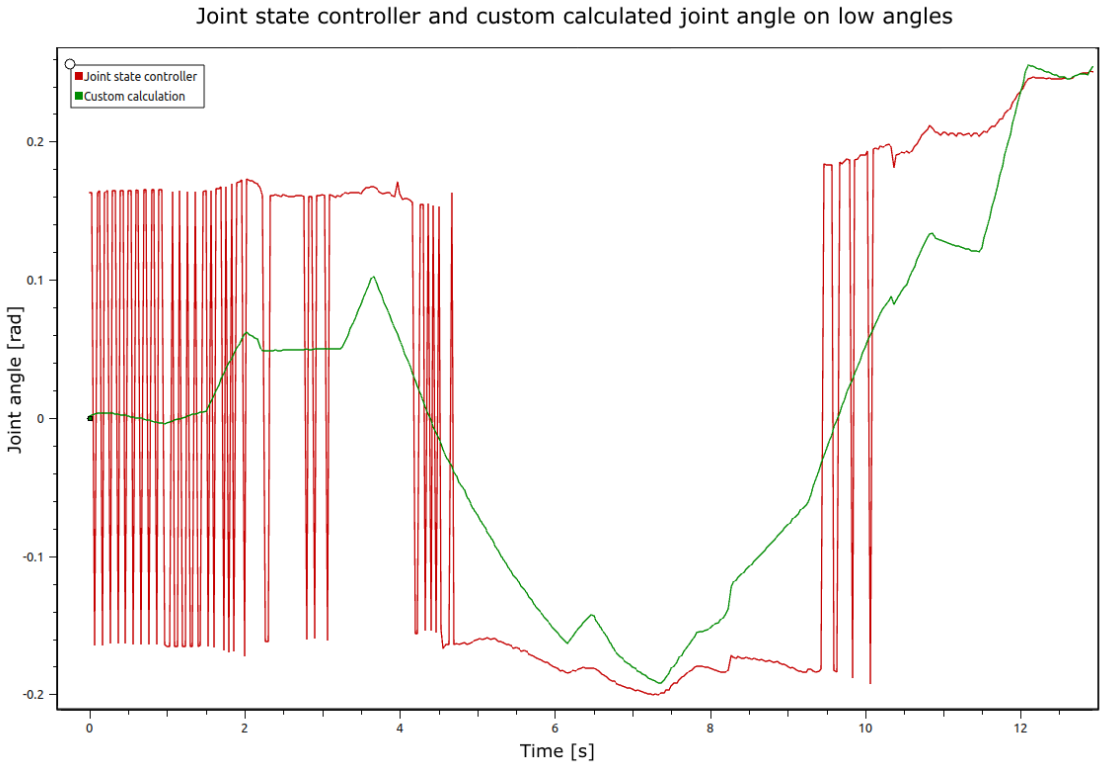

Figure 10. Joint state controller and custom calculated angle comparison. Angle estimates for joint state controller were not aligned with visualisation from simulation, while custom calculations provided a better match.

3.4.2 Modeling the whisker sensor

The spring stiffness of the whisker was calculated using 8 random whisker sensors and a set of weights tied to the ends of the whiskers at a known distance (Figure 11). The whisker sensors were fixed to a position parallel to the ground, interfaced with a computer via a USB port and their angle, calculated from magnetic field strength, was measured at a 30 Hz frequency. The angle was then measured eight times with different amounts of

weight tied to the ends of the whiskers. From this setup we can approximate the stiffness of the whisker using Hooke's law with:

$$k = -\frac{m\ g\ \ l_2\ \cos x}{dx\ l_1},$$

where k is the whisker stiffness, m is mass tied to the end of the whisker, g is gravitational acceleration, dx is the angular displacement caused by the weight, and x is the inclination angle reported by the whisker, l_2 is the distance from mass to whisker attachment point, and l_1 is the distance from whisker attachment point to the flexible base (Figure 12). Hooke's law is a simplification of the actual physical process, which was used, because Gazebo uses the spring stiffness constant to model elasticity of the joint. Averaged results of this experiment can be seen on Figure 13.

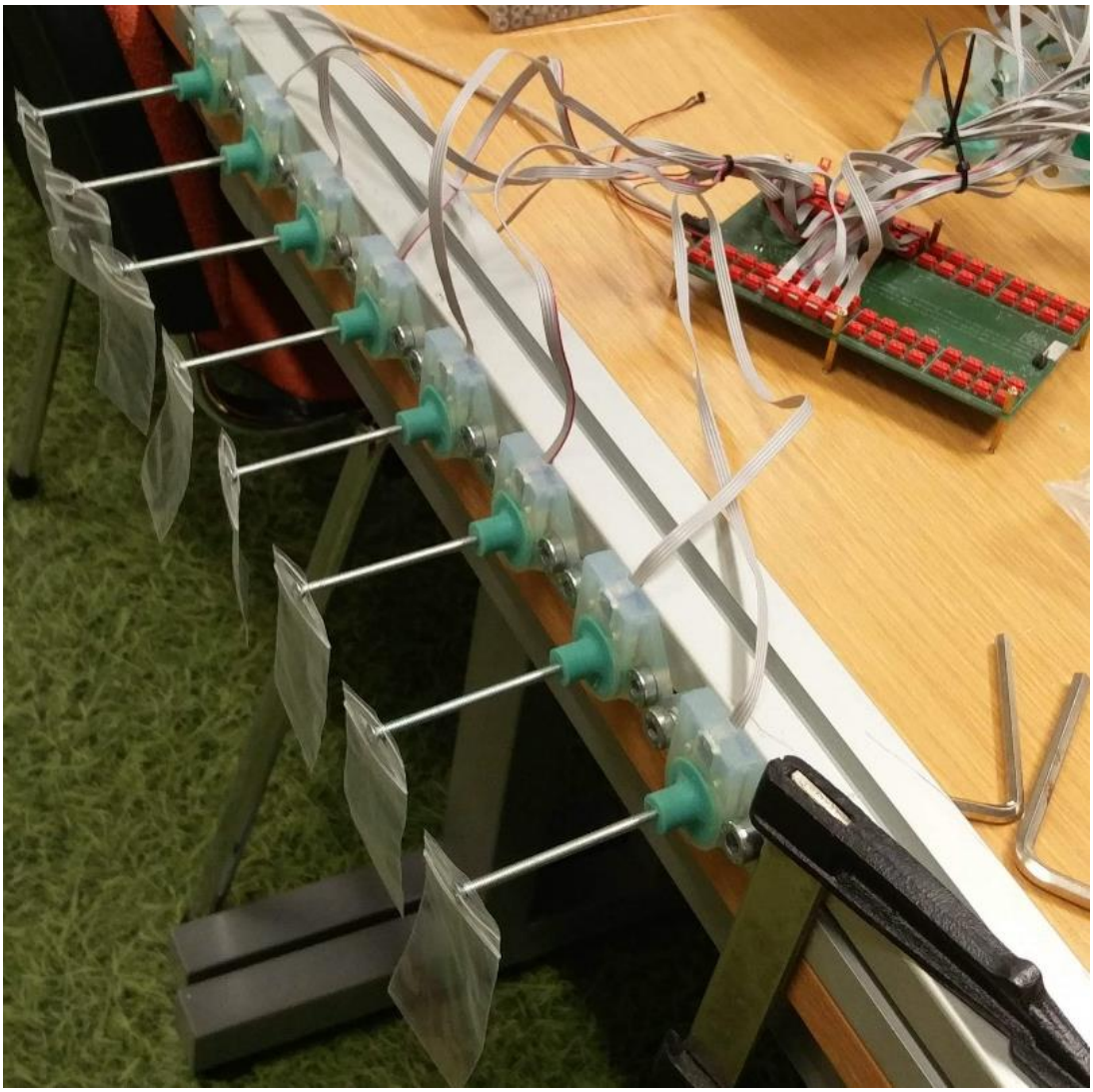

Figure 11. Whisker base stiffness measuring setup.

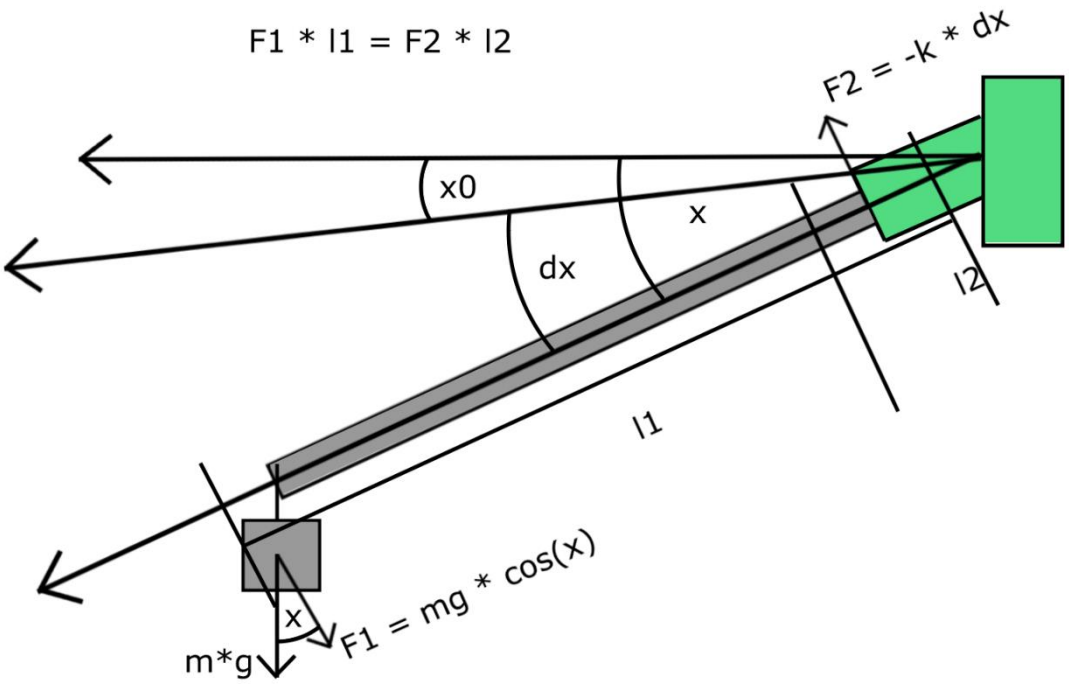

Figure 12. Schematics for whisker stiffness measuring setup.

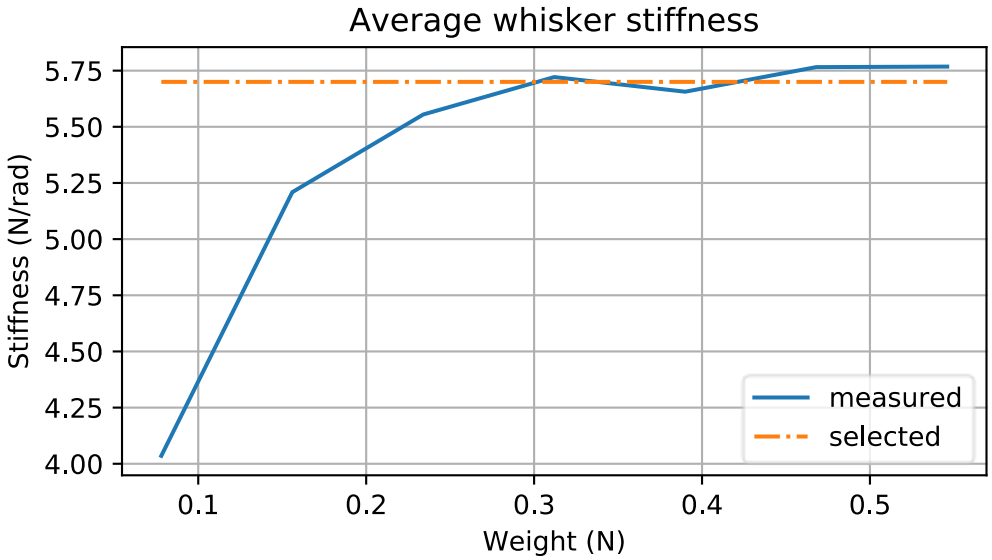

Figure 13. Averaged whisker stiffness of eight whiskers over eight measuring points with the approximated stiffness constant inserted to the simulation.

Using the measurements for each whisker a stiffness to force plot was produced. Using this plot, the stiffness was estimated to be approximately 5.7 N/rad.

The other dynamic elasticity parameter required by Gazebo, the viscous damping coefficient, was estimated by measuring the amplitudes of the whisker oscillation between two consecutive oscillations and estimating the logarithmic decrement. The measurements were taken at 30 Hz and did not yield an accurate sinusoidal graph of the

whisker oscillations due to the relatively high oscillation frequency of the whisker sensor and low sampling rate. Due to that issue, the logarithmic decrement was estimated relatively roughly from the measurements, which still allows to get some estimation for the viscous damping coefficient.

The viscous damping coefficient *c* was found with the equation:

$$c = \zeta \sqrt{m\,k},$$

where ζ is the damping ratio of the spring, *m* denotes the mass attached to the spring and k is the spring stiffness constant. The damping ratio was found by using the logarithmic decrement δ:

$$\zeta = \frac{\delta}{\sqrt{\delta^2 + (2\pi)^2}},$$

where the logarithmic decrement was estimated from measurements to be approximately $\delta = 3.35$. The damping ratio can then be estimated as $\zeta = 0.47$. With $m = 0.008\ kg$ and $k = 5.7\,\frac{N}{rad}$, the viscous damping coefficient was estimated to be $c = 0.1\,\frac{N\,m\,s}{rad}$.

3.4.3 Vehicle framework

The platform was moved in the simulation by using *gazebo_ros_force_based_move* plugin from hector_gazebo [20]. The planar movement plugin included with Gazebo ROS packages, updates a link's velocity in simulation, which led to high accelerations when changing velocity. This in turn led to unpredictable behaviour in simulation during the development of this thesis. The force-based move plugin however, takes a velocity input and updates forces acting on the link in simulation leading to smoother acceleration and more predictable behaviour.

To give velocity commands to the planar movement plugin a waypoint controller was developed, which takes a list of waypoints and outputs velocity commands until the robot reaches the designated waypoint. To ensure that the controller reaches a designated waypoint, it listens to robot's ground truth position from Gazebo. Each waypoint consists of x-coordinate, y-coordinate and heading direction. The array can be dynamically appended to during runtime via the Rviz visualisation tool. The controller was used to

ensure similar movement in each simulated test run, which would not be possible with a human operator, while also giving the option to manually move to robot if needed.

For the SLAM algorithm to behave as it would in a real environment, the biased odometry ROS node was also developed. It is currently difficult to predict the odometry errors from the 4-screw drive prototype, as it has not seen enough development and testing. For the evaluation of the SLAM algorithm, the error model of the odometry also does not have to match the one on the final robot exactly. Therefore, the odometry model added to the simulation framework was made considering only the degrees of freedom in locomotion, that the robot will have – the heading angle and the linear motion in x and y axis. Therefore, the robot's pose here is described by values [*x, y, θ*], which denote the x-coordinate, the y-coordinate, and the heading angle from x-coordinate respectively. The x-, and y-coordinate of the robot can be described in either the world frame, or the robot frame. The world frame coincides with the robot frame at the start of the simulation but is fixed with respect to the environment objects. The robot frame coincides with the sensor grid's centre of mass, is fixed with respect to it, and moves with it.

Using its previous odometry estimate as robot's pose estimate, the odometry node calculates the robot's velocity with the following:

$$v_x^{gt} = \frac{s_x^{gt}}{t}$$

$$v_y^{gt} = \frac{s_y^{gt}}{t}$$

$$v_\theta^{gt} = \frac{s_\theta^{gt}}{t}$$

where s_i^{gt} denotes the ground truth displacement from last iteration for $i \in \{x, y, \theta\}$, v_i^{gt} denotes the ground truth velocity for $i \in \{x, y, \theta\}$, and t is time step from last iteration.

Then robot's velocity is transformed to robot frame with:

$$\begin{bmatrix} v_x^{gtr} \\ v_y^{gtr} \\ v_\theta^{gtr} \end{bmatrix} = \begin{bmatrix} \cos(\theta^{gt}) & -\sin(\theta^{gt}) & 0 \\ \sin(\theta^{gt}) & \cos(\theta^{gt}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x^{gt} \\ v_y^{gt} \\ v_\theta^{gt} \end{bmatrix}$$

where θ^{gt} denotes ground truth robot rotation angle in world frame, and v_i^{gtr} denotes robot's ground truth velocity in robot frame for i ∈ {x, y, θ}.

As the next step, gaussian noise with non-zero mean is added to simulate incorrect odometry estimation. Each dimension can have different noise characteristics defined by the mean of noise μ_i, and standard deviation of noise σ_i for i ∈ {x, y, θ}.

$$v_x^{odomr} = v_x^{gtr} + G(\mu_x, \sigma_x)$$

$$v_y^{odomr} = v_y^{gtr} + G(\mu_y, \sigma_y)$$

$$v_\theta^{odomr} = v_\theta^{gtr} + G(\mu_\theta, \sigma_\theta)$$

In the equation v_i^{odomr} stands for robot's odometry velocity estimation in robot's frame for i ∈ {x, y, θ}, and $G(\mu, \sigma)$ represents a sample from gaussian distribution with mean μ and standard deviation σ.

Finally, the robot's odometry velocity estimations are transformed to world frame with:

$$\begin{bmatrix} v_x^{odom} \\ v_y^{odom} \\ v_\theta^{odom} \end{bmatrix} = \begin{bmatrix} \cos(-\theta^{odom}) & -\sin(-\theta^{odom}) & 0 \\ \sin(-\theta^{odom}) & \cos(-\theta^{odom}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x^{odomr} \\ v_y^{odomr} \\ v_\theta^{odomr} \end{bmatrix}$$

Where θ^{odom} denotes odometry estimation of robot's heading angle, and v_i^{odom} stands for robot's velocity estimation in world frame for i ∈ {x, y, θ}.

These odometry estimates of the robot's velocities are then used to calculate the robot's position and rotation estimates in world frame.

3.4.4 SLAM

The SLAM algorithm from [17] consists of four modules – prefiltering, scan matching odometry, floor detection nodelet, and graph-SLAM nodelet (Figure 14). The prefiltering module needs a point cloud measurement input, typically from a lidar, and the graph-SLAM module outputs a map estimation, together with a pose correction estimate. The modules use Point Cloud Library [21] implementation of point clouds.

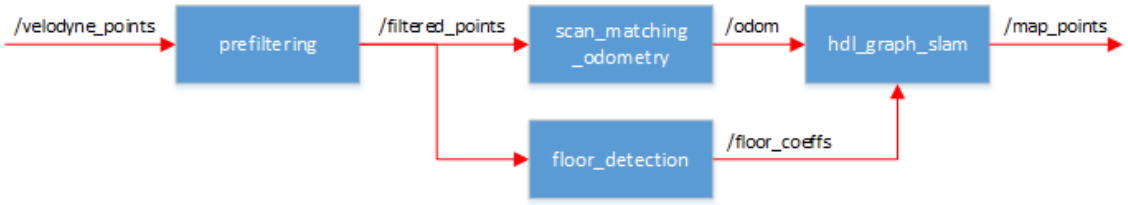

Figure 14. The graph SLAM module flowchart.[22]

For the purposes of this thesis the floor detection module was removed and scan matching odometry input to graph SLAM was replaced with biased odometry created by the author. The input for prefiltering node was the point cloud generated by the whisker sensor grid measurements.

The prefiltering node takes in a pointcloud measurement and filters it according to selected criteria. Filtering allows to remove outliers and downsample the point cloud if necessary. Outlier removal should increase the quality of matchings for both scan mathing odometry and the graph-SLAM node. Downsampling the point cloud allows to reduce the number of points in a densely scanned region and thus reduce the complexity of scan matchings. During this thesis only downsampling was used, due to the reason that outlier detection techniques for the whisker sensor grid have not yet been evaluated. The downsampling was performed using the *VoxelGrid* approximation from PCL [21] library.

The scan matching odometry node was removed from the pipeline to simplify initial implementation and testing. The whisker point cloud is relatively small and sparse compared to a usual lidar. Therefore, the scan matching odometry is unlikely to perform well without modification and using it would increase the complexity of evaluating and testing the graph-SLAM algorithm.

Similarly, the floor-detection node is unnecessary for the whisker sensor grid. The sensor in its current stage is intended to map only the floor, therefore having an additional floor-detection algorithm running on the sensor measurements is not considered beneficial.

The graph-SLAM algorithm initially worked by taking a scan from lidar, comparing this to previously taken scans in a radius around the current position estimate if the conditions are met, and adding the scan to the map. A fitness score is calculated for a matching between the new scan and scans in the map. If the fitness score clears a threshold, then a

constraint is added between the two scan poses in the pose graph, which represents a loop closure. This works for lidar, because the scans are dense and cover a large area compared to the whisker sensor. The condition for trying a new matching is distance travelled in x or y dimension from last matching.

For use with the whisker sensor the algorithm was modified (Figure 15). Instead of taking a single scan, the scans are cumulatively added to a point cloud until a criterion is reached, forming a keyframe, which is added to the pose graph. Adding these scans to the composite scan is done based on odometry pose estimate. The criterion to add a composite scan to the map is fulfilled when, either the sensor position change in x or y dimension, or the heading angle change, crosses a threshold. This composite scan is matched with other composite scans in the map in the same manner as before.

If the composite scan is completed over a large distance, then it will be incorrect due to the odometry drift. This in turn makes it harder for the graph-SLAM algorithm to correct the map. During this thesis, a value of 0.12 meters for linear displacement and 1.54 radians for angular motion was used as a threshold for completing the composite scan. This gave a large enough scan while not accruing too much inaccuracy in the point cloud due to odometry drift.

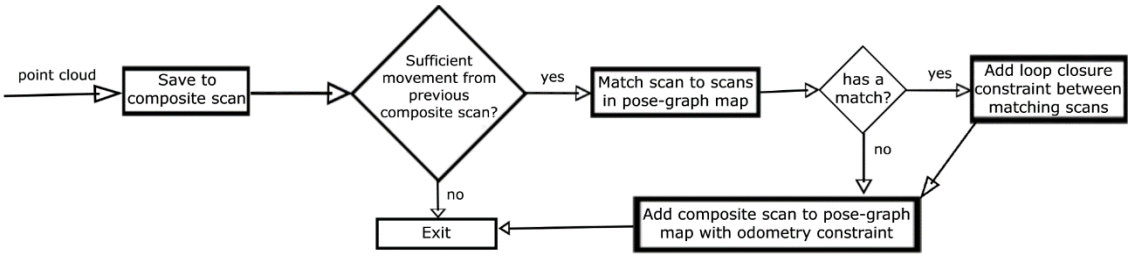

Figure 15. Flowchart of the modified SLAM algorithm used with whisker sensor grid.

The scan matchings in the graph-SLAM algorithm [17] are performed using the PCL [21]. While using the PCL point cloud implementation and matching methods itis important to choose a point cloud registration resolution and point cloud matching algorithm from the ones implemented in the library. During the development NDT [23], GICP [24], and FAST_GICP[25] implementation were tested. The NDT and GICP implementations were from PCL, while FAST_GICP implementation was based on [25], [26]. The matching algorithm chosen was FAST_GICP and the resolution for point clouds was set to 0.1 meters, as these settings gave good SLAM performance on visual inspection during

development. The fitness score threshold was set to 0.016 from, which seemed to give small number of incorrect loop closures, while still maintaining the ability to perform correct loop closures. To limit the possible matching options and increase algorithm speed, the maximum keyframe matching radius was set to 0.75 meters. This still allowed to correct for large odometry errors, while speeding up the scan matching loop closure calculations, by decreasing the number of scans, for which the fitness score was necessary to calculate. It is entirely possible, that other settings can give similar or even better performance, as comprehensive parameter tuning, based on empirical evidence was not performed, to limit the scope of the thesis.

4 Results and Discussion

This chapter presents the results obtained by the techniques described in the previous chapter and analysis of the said results. In section 4.1 the results of localisation and mapping are presented for individual test scenarios described in 3.3, as well as the comparison between the scenarios. Section 4.1.6 shows the repeatability of experiments on the basis of scenario 3. In addition, the options for future improvements of the SLAM algorithm and whisker grid sensor are highlighted.

4.1 SLAM

The results from individual test scenarios are presented in this section. From Figure 17 to Figure 27 we can see the maps and trajectories resulting from the SLAM algorithm, as well as the actual trajectory followed by the robot in the simulation. The visualisation of the actual map in simulation is visible in Figure 8, with a side view perspective visible in Figure 5. Visually, maps generated from scenario 1-4 look similar to the ground truth map and the artificial objects' locations can be seen. In addition, the uneven terrain features can also be seen. Visual inspection of the figures suggests that the SLAM algorithm performed well in scenarios 1 to 4.

On Figure 16 the localisation results of odometry and the SLAM algorithm are represented side by side. The SLAM algorithm does manage to correct the localisation estimate on all five scenarios, improving on the localisation estimates given by odometry. The odometry estimate starts out accurate but grows unbounded. The localisation estimate for this SLAM algorithm behaves similarly unless it reaches a previously known area and recognizes it. Recognizing the location allows to correct the localisation estimate and bound the error estimate.

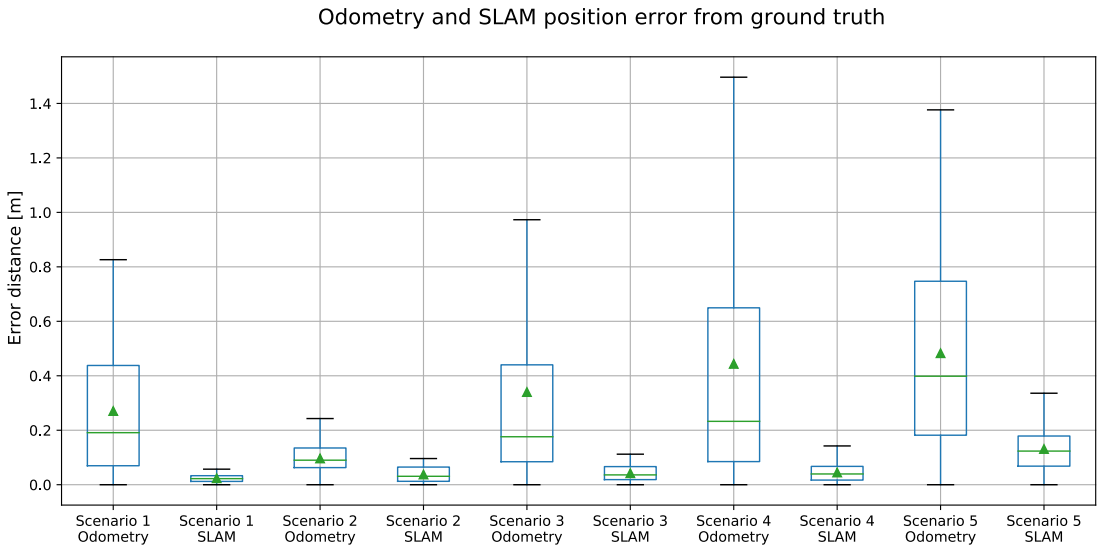

Figure 16. Boxplots of SLAM and odometry errors from ground truth during all 5 driving scenarios. Mean error denoted with green triangle. Error distance calculated as Euclidean distance from x and y coordinates of robot.

4.1.1 Scenario 1

During scenario 1, the sensor array performed multiple back and forth passes over the same environment, while rotating at the ends of the passes. It can be observed that the odometry-based localisation estimate deteriorates with every pass. On the first half-length pass, the odometry stays quite accurate, with localisation error of less than 0.1 meters visible on path (Figure 18). During the full-length pass, the odometry error continues to accumulate, while localisation error for SLAM algorithm stays low. This behaviour continues with each pass, suggesting that the SLAM algorithm keeps the localisation error bounded. We can see the SLAM algorithm recovering from incorrect pose estimations where the yellow path goes through a sharp sideways motion, which is not visible on ground truth and odometry trajectories.

The map from scenario 1 (Figure 17) is visually similar to the ground truth map. The brick, the slide, and the stone are all clearly visible in purple colour, which denotes higher locations and distinguishes them from their surroundings. The pipe is visible as a blue streak with slight purple dots. The hemisphere is visible as a circular relatively unmapped shape on the right side of the mapped area, with some higher points visible on it. This is because the whiskers tend to follow the lowest areas on the terrain, due to gravitation and the spring effect of the sensor, and therefore slip off the sphere without mapping the top part of it. The points visible on the sphere are often mapped due to the interactions of

whiskers with each other, which sometimes force some of the whiskers off the lowest terrain.

The uneven terrain features also become visible, as the higher location at the centre of the map is coloured purple and lower locations on either of the sides are coloured orange. The fact that higher and lower objects are both of the same colour on either end also indicates that the map is parallel to the xy-plane. Here similar behaviour is visible while mapping the sphere - some locations in the middle of the mapped area are not mapped. This is due to the robot following exactly the same path over each pass, and the whiskers following the lower contours of the terrain, therefore not mapping higher locations from which they can slide off. Overall, the mapping result from scenario 1 is in scale and accurate.

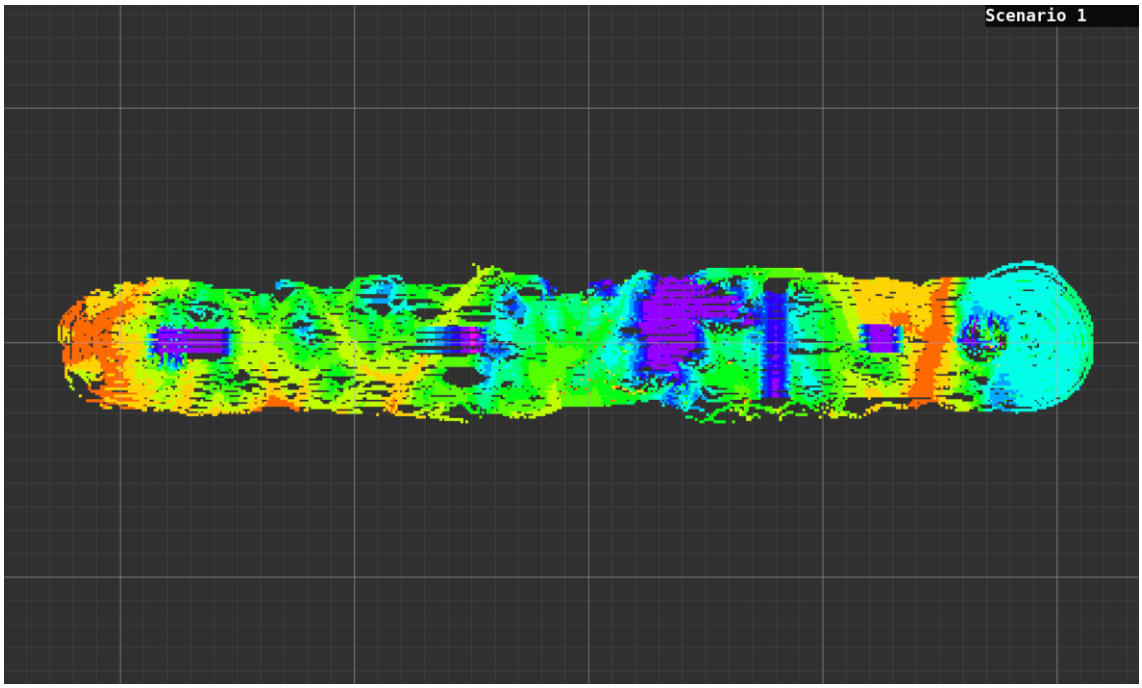

Figure 17. Mapping results visualized as a height map from scenario 1. Violet/blue - highest points, orange/red - lowest points. Grid size and axes same as before.

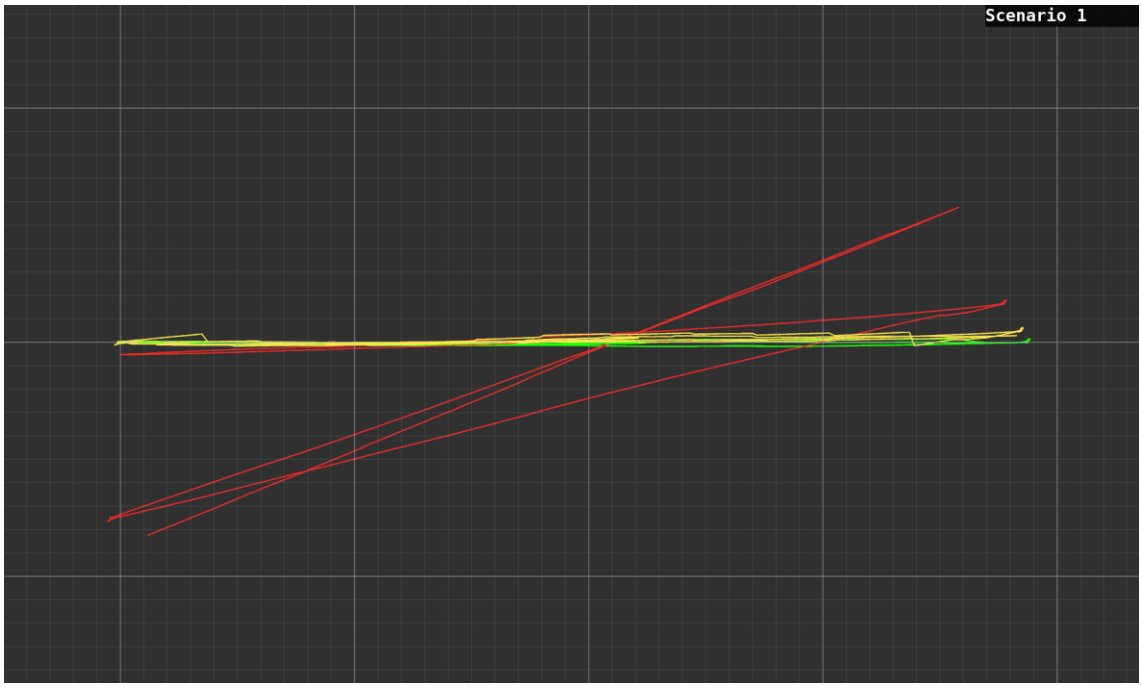

Figure 18. Path travelled by robot in first scenario. Green - ground truth, red - odometry, yellow - SLAM algorithm.

4.1.2 Scenario 2

In the second scenario the robot follows a small circular trajectory performing two passes (Figure 20). During the first pass, the odometry and SLAM algorithm estimates coincide, leaving only the SLAM path visible. When the robot reaches the start of the circle again, the SLAM algorithm corrects its pose estimate closer to ground truth while odometry error continues to accumulate. The pose estimate is most visible in the ending phase, when robot returns to its starting location, where SLAM pose estimate mostly coincides with the ground truth, while odometry does not.

The mapped area from the second scenario (Figure 19) should contain two cuboids (‘brick’ and ‘slide’), with the one closer to the starting point being level and the other one slanted. The brick is clearly visible on the left side of the map. The slide is barely visible just across the small loop, as a rough rectangular shape of blue dots. The slide is this time less clear, due to the fact that unlike the previous scenario, it is being approached from the side and the whiskers slide off of it, instead of crossing. This more clearly highlights the behaviour noted during the analysis of scenario one, that the whiskers mostly gather points from locally lower areas of the terrain.

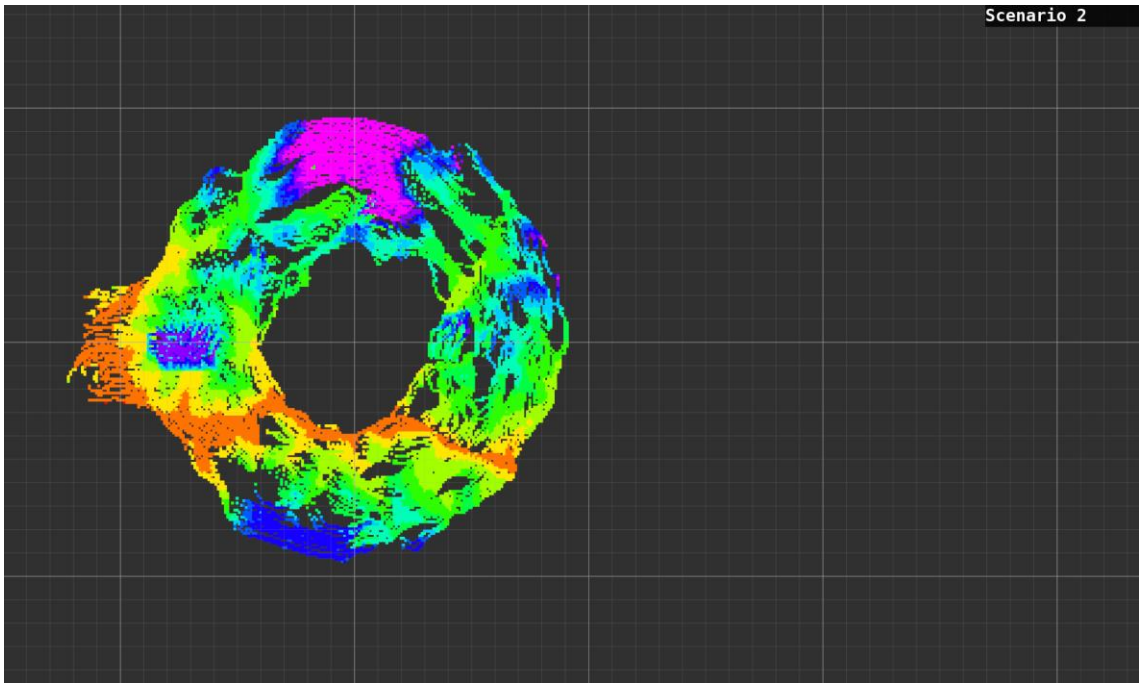

Figure 19. Mapping results visualized as a height map from scenario 2. Violet/blue - highest points, orange/red - lowest points.

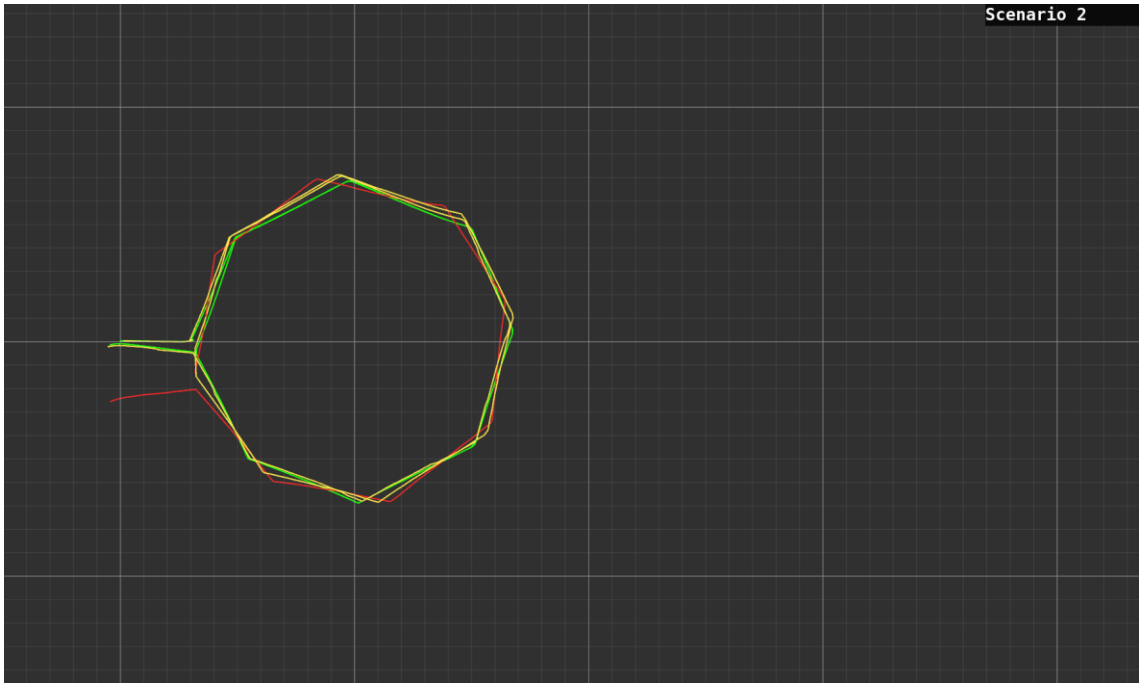

Figure 20. Path travelled by robot during scenario 2. Green - ground truth, red - odometry, yellow - SLAM algorithm.

4.1.3 Scenario 3

The third scenario consisted of scenario 2, followed by scenario 1. The same behaviour noted in scenario 2 is visible, where SLAM algorithm manages to correct its pose estimate during the second small loop (Figure 22). The following movement seems to perform similarly to scenario 1, with SLAM algorithm continuing correctly over the already

mapped area. Approaching the already mapped area from a different direction could be an issue for the SLAM algorithm. This is due to at least two effects: first, the mapped area will be slightly different due to the whiskers following the lowest curves. And second, using the tip-contact assumption mentioned in section 3.1, the mapped points from previous pass will be in slightly different positions when approaching from a different direction for higher and steeply rising objects, such as the cuboid. It can be seen by comparing the maps from first, second, and third scenario where mapped areas intersect – for example, the region around the slide (Figure 17, Figure 19, Figure 21). However, it does not seem to cause an issue for the sensor here and the resulting map is still accurate and in scale (Figure 21).

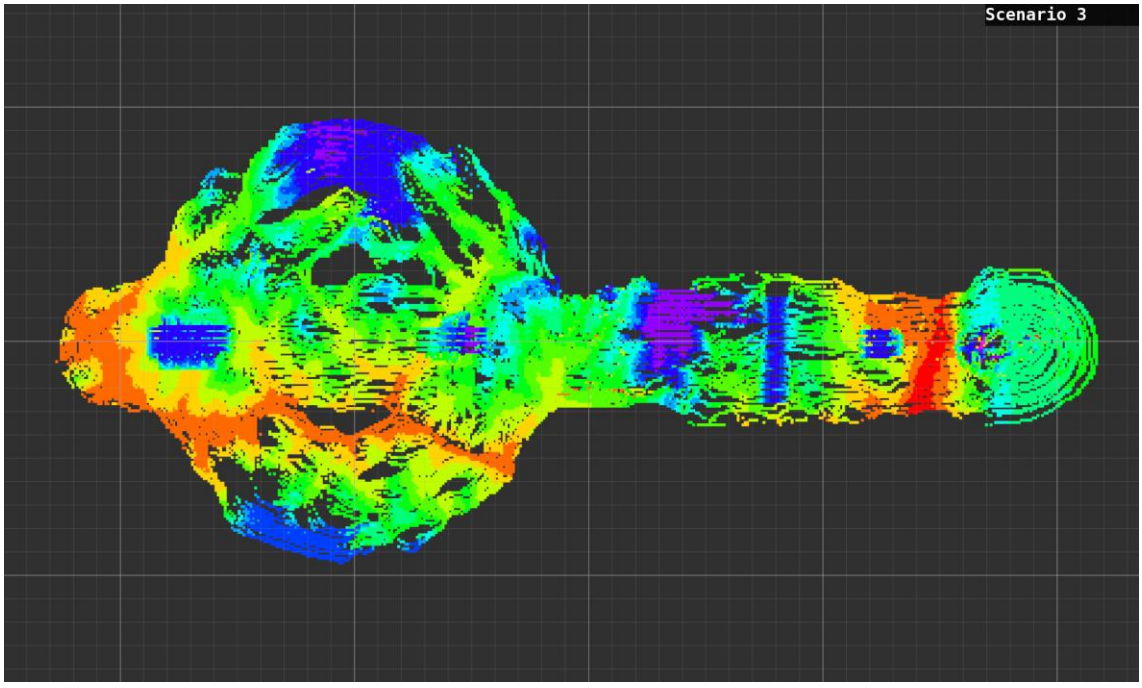

Figure 21. Mapping results visualized as a height map from scenario 3. Violet/blue - highest points, orange/red - lowest points.

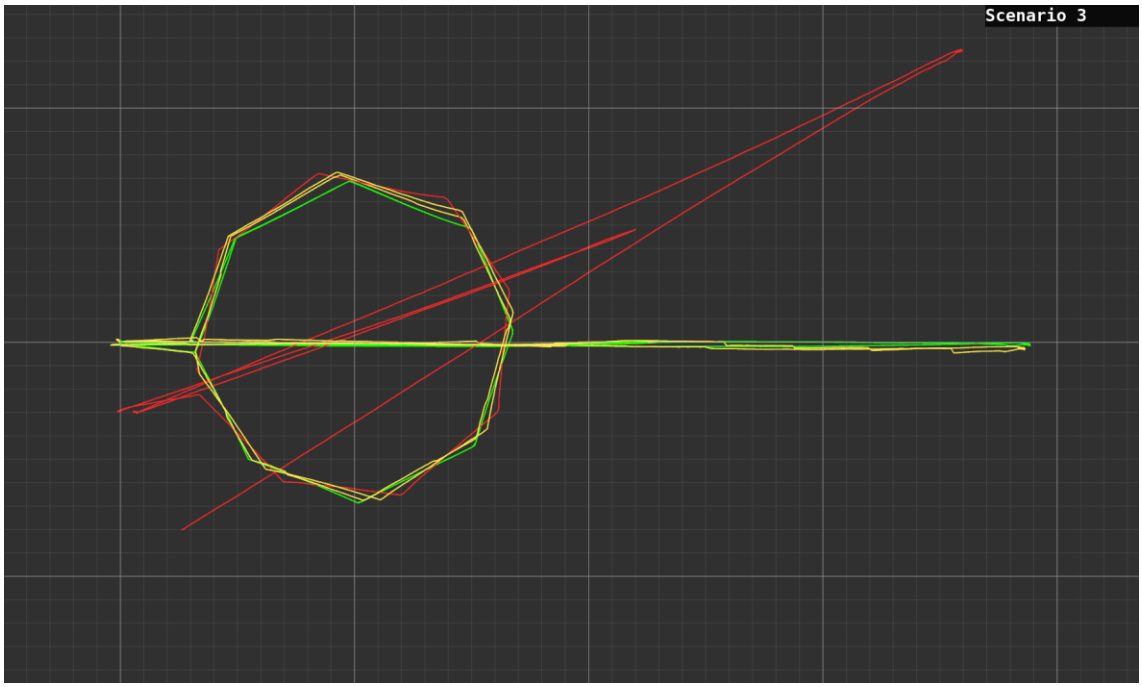

Figure 22. Path travelled by robot during scenario 3. Green - ground truth, red - odometry, yellow - SLAM algorithm.

4.1.4 Scenario 4

From scenario 4 we can see that the SLAM algorithm starts to have slight issues. On the large loop the pose estimate gets less accurate, and upon reaching a previously mapped area the localisation error starts diminishing. When mapping the area between points S and A, there seems to have been relatively large localisation errors in yaw angle, where the SLAM pose estimation starts to drift from ground truth (Figure 24). These localisation errors are the result of adding an incorrect constraint edge to the pose graph, which can lead to issues in continuing operation. In this case it seems, that following localisation attempts manage to correct the robot pose and the pose graph as well (Figure 25), because the mapping result looks relatively similar to the ground truth map. The edges on the pose graph consist of odometry edges, which should coincide with the actual trajectory of the robot and loop closure edges, which are added when the algorithm recognizes a previously visited location.

Looking closer at the mapping result (Figure 23) all five artificial features can be recognized. The mapped area between waypoints A and B is this time mapped more sparsely. This is probably due to the robot only passing this area from one direction. This effect can be seen on the right side of the stone. The area opposite from the robot's approaching direction, right behind the stone, is mapped very sparsely, while the area in

front of cuboid is mapped densely. When moving over the stone, the whiskers fall off of it very quickly, leaving part of the terrain unmapped. While climbing the cube, on the other hand, the whiskers map many points at almost the same xy-coordinate but at different height.

It can be seen that the scenario 4 map is slightly crooked, the hemisphere is not exactly in its actual location and points on the left side of the map tend to be slightly higher than on the right side of the map. This is probably due to slightly incorrect loop closures performed after the large loop. Overall, the map is still fairly accurate and would perform well for navigation over the mapped area.

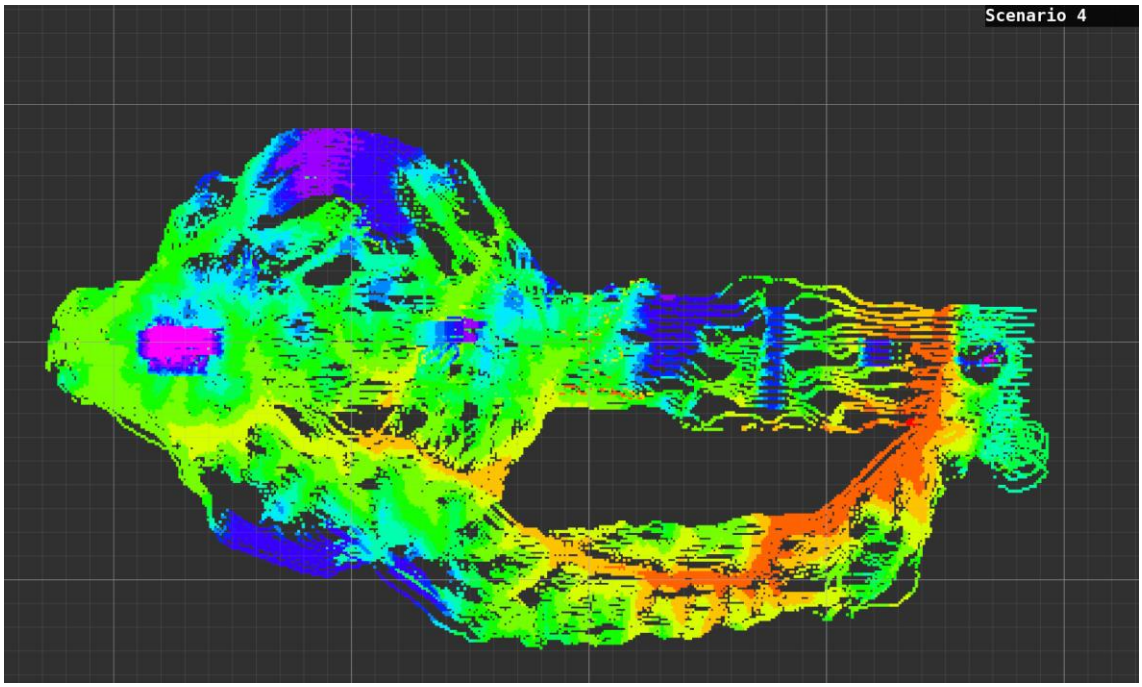

Figure 23. Mapping results visualized as a height map from scenario 4. Violet/blue - highest points, orange/red - lowest points.

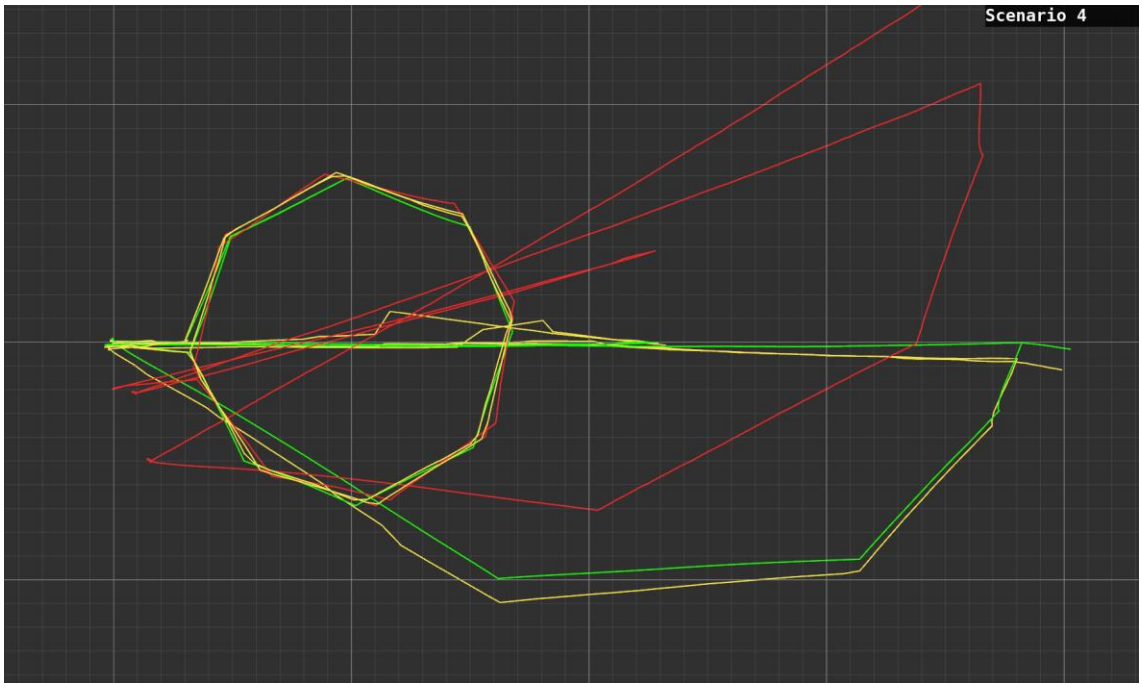

Figure 24. Path travelled by robot during scenario 4. Green - ground truth, red - odometry, yellow - SLAM algorithm.

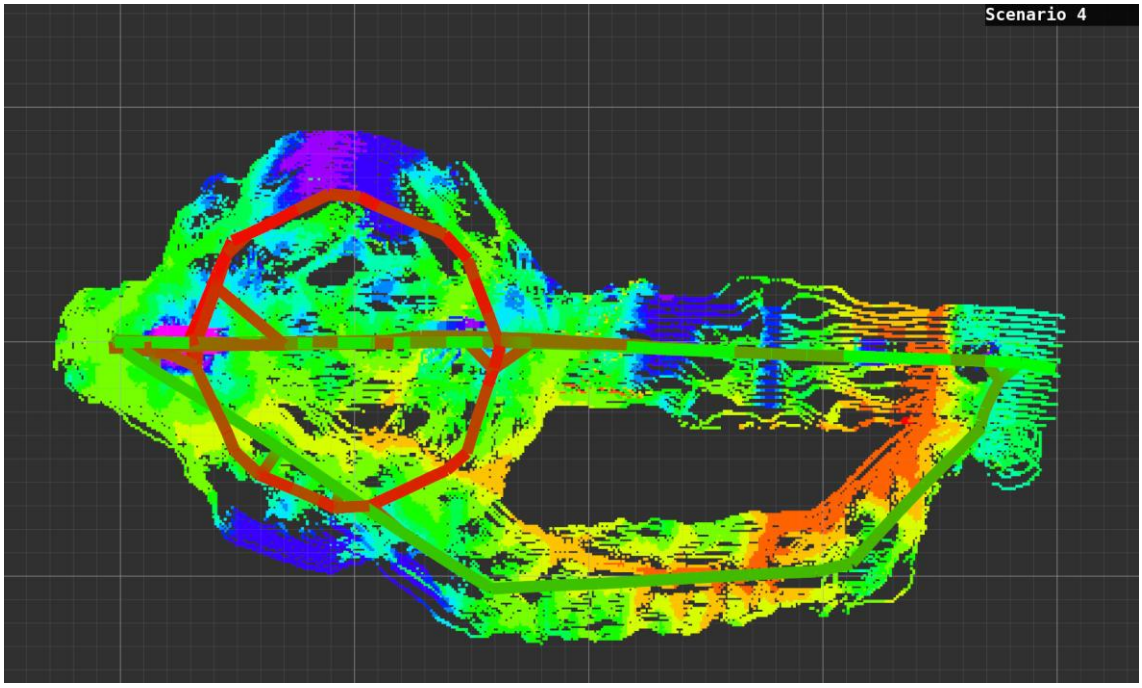

Figure 25. Scenario 4 SLAM map with pose graph.

4.1.5 Scenario 5

On Figure 27 the path of the whisker sensor in scenario 5 can be seen. The SLAM algorithm seems to be stuck on the same path on each of the large loops, that it was travelling during the variation of the large loop. This could mean that it does not recognize the ground features which have been mapped from different locations. To clarify, we can

look on Figure 28, where the pose graph of SLAM algorithm has been visualised with the resulting map. It can be seen there that the pose graph consists of at least two variations of the large loop, as it should. Therefore, the SLAM algorithm seems to perform fairly well even on the largest scenario according to sensor trajectory information.

On the map from scenario 5 (Figure 26) all five artificial terrain features can once again be recognized. Some issues can also be spotted – the map is rotated counter-clockwise by a small angle and the brick looks slightly spread out on the map. The map is also not parallel to the xy-plane – it seems to descend towards the top right part of the map. These issues stem from the robot not recognizing its starting location after performing the large loop with enough accuracy. Looking at the pose graph, two sharp turns in odometry edges can be seen right after leaving starting position S. Two loop closure edges can also be seen, connecting the starting location to the position reached after the large loop. These seem to indicate, that the algorithm added two incorrect constraints when it reached the starting location after performing the large loop. Continuing on the second large loop, the algorithm adds more correct constraints between the large loops it is travelling on and manages to continue mapping. While performing the small loops it continues to add correct loop closures but does not manage to correct its mapping of the starting location. This indicates the risks of adding incorrect constraints to the pose graph – once incorrect localisations are made, they can be difficult to overcome by the algorithm. Overall, the map is still recognizable and mostly in scale.

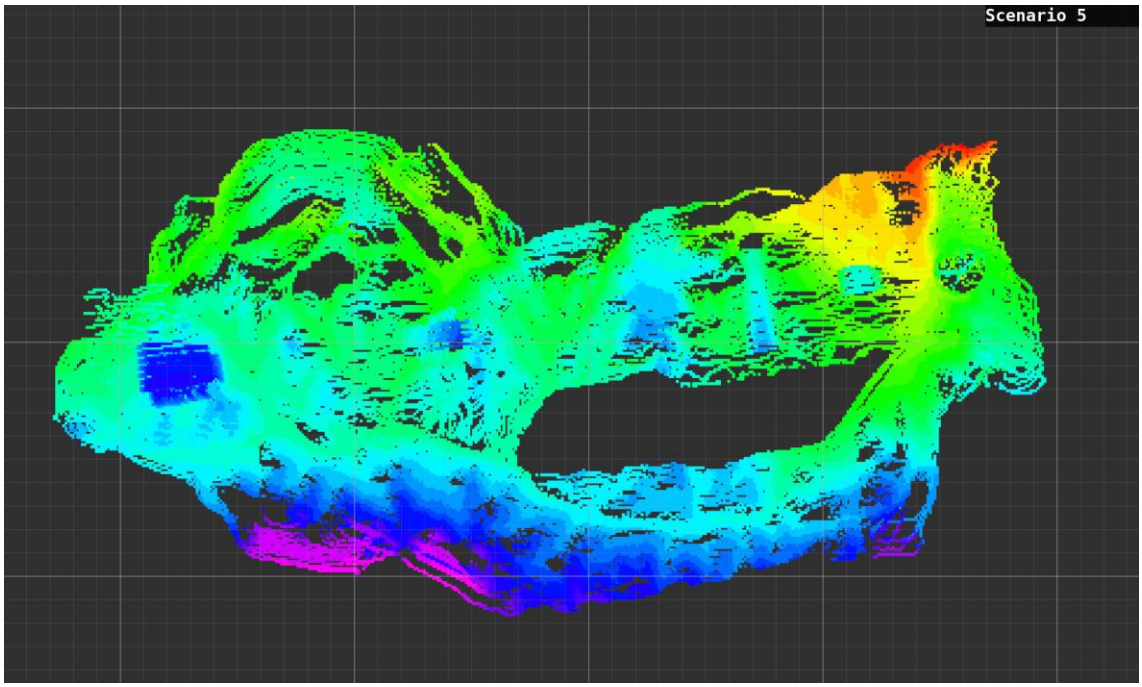

Figure 26. Mapping results visualized as a height map from scenario 5. Violet/blue - highest points, orange/red - lowest points.

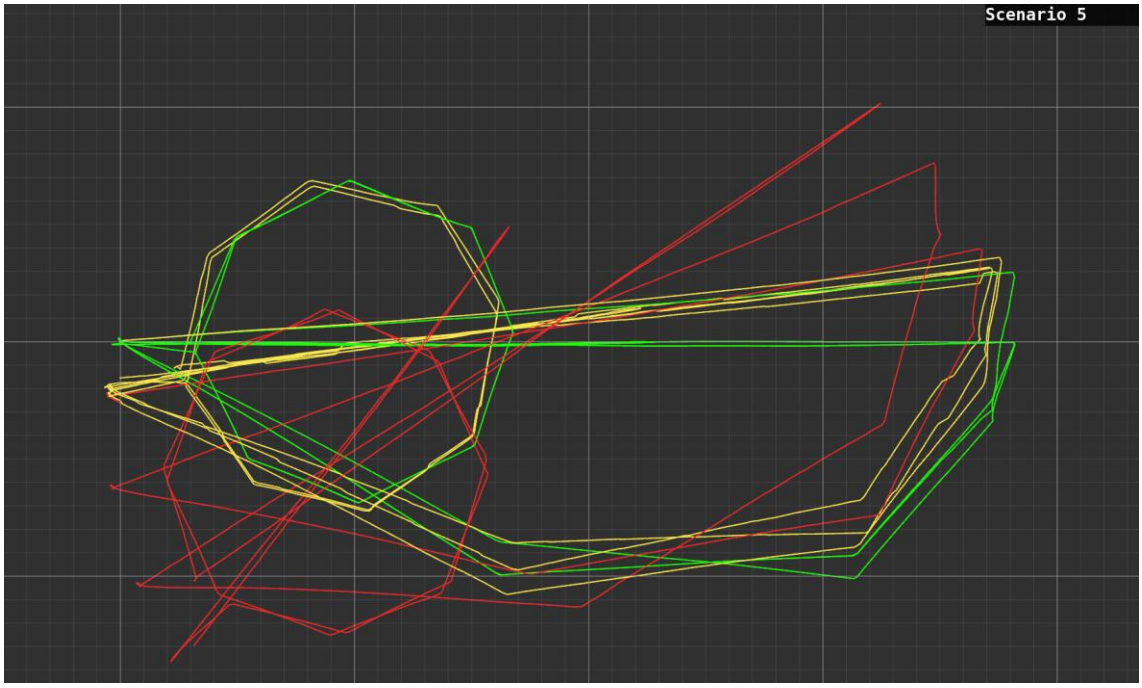

Figure 27. Path travelled by robot during scenario 5. Green - ground truth, red - odometry, yellow - SLAM algorithm.

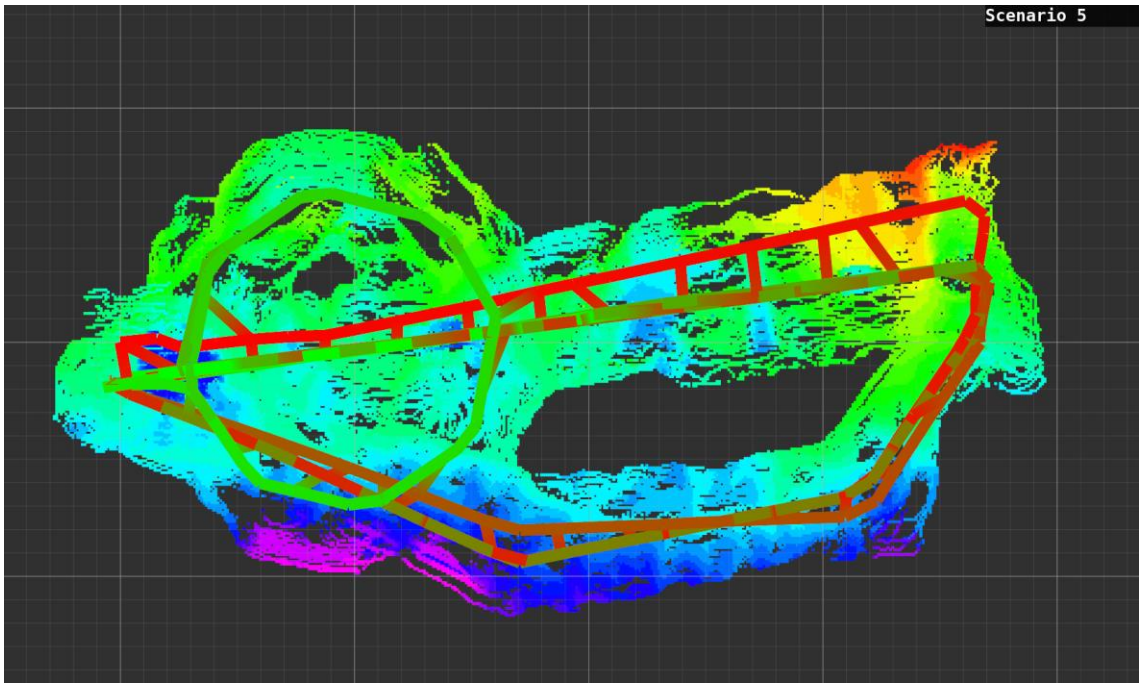

Figure 28. Mapping results from scenario 5 with pose graph edges visible.

4.1.6 SLAM repeatability analysis

To give insight into the repeatability of this SLAM process, 10 runs of SLAM were conducted on the same recorded dataset of the simulation with the same SLAM parameters. The simulated dataset chosen for repeatability tests originates from scenario 3. The results are numerically presented on Figure 29 with a visualisation of all mapping results on Figure 30.

Although the dataset recorded from simulation is identical for each run, the results of the SLAM algorithm pose estimation can differ due to gaussian noise in odometry estimation, or timing differences caused by differing workload of the computer. The odometry was not recorded, but generated on the recorded dataset, to ensure the robustness of the algorithm and is thus the most probable cause for small changes in the results. However,

small initial changes in the pose graph map of two identical runs can introduce larger changes later on.

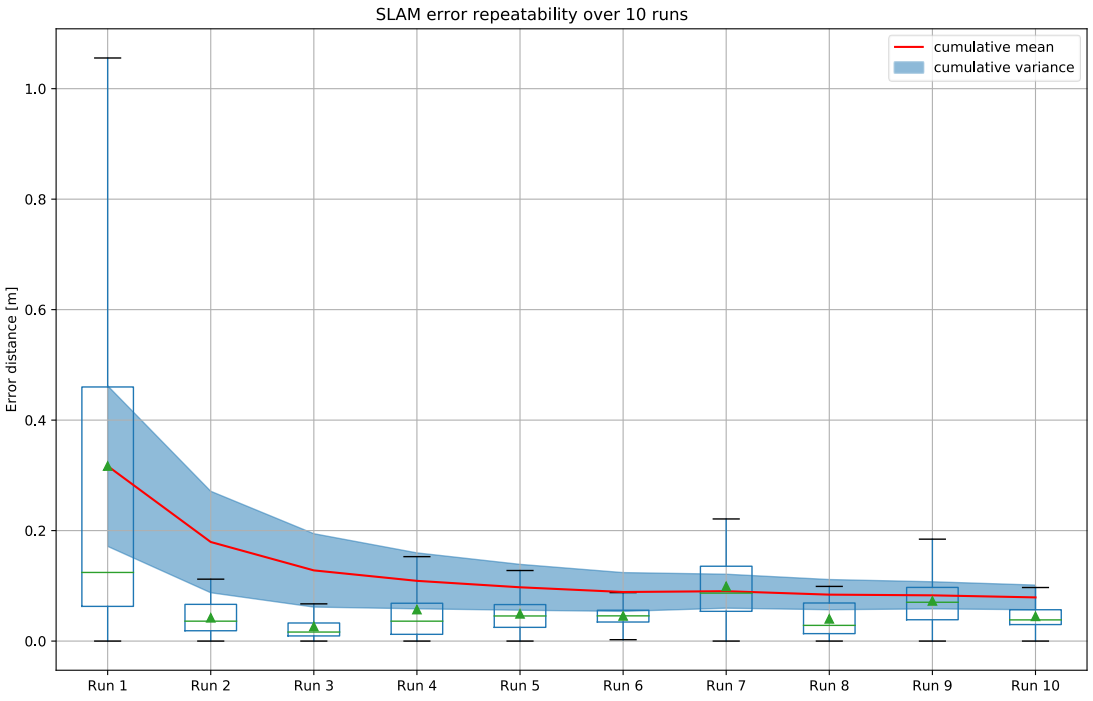

Figure 29. Boxplot of SLAM errors over 10 tests over same scenario. Cumulative mean error takes into account all mean errors from current and its preceding runs.

On Figure 29 it can be seen that the errors of the localisation estimate deviate from run to run. For 9 out of 10 runs the mean and median error of the run stay below 0.2 metres. The cumulative mean error and variance of the localisation error also seem to converge, which can be interpreted as a sign, that the experiments are generally repeatable.

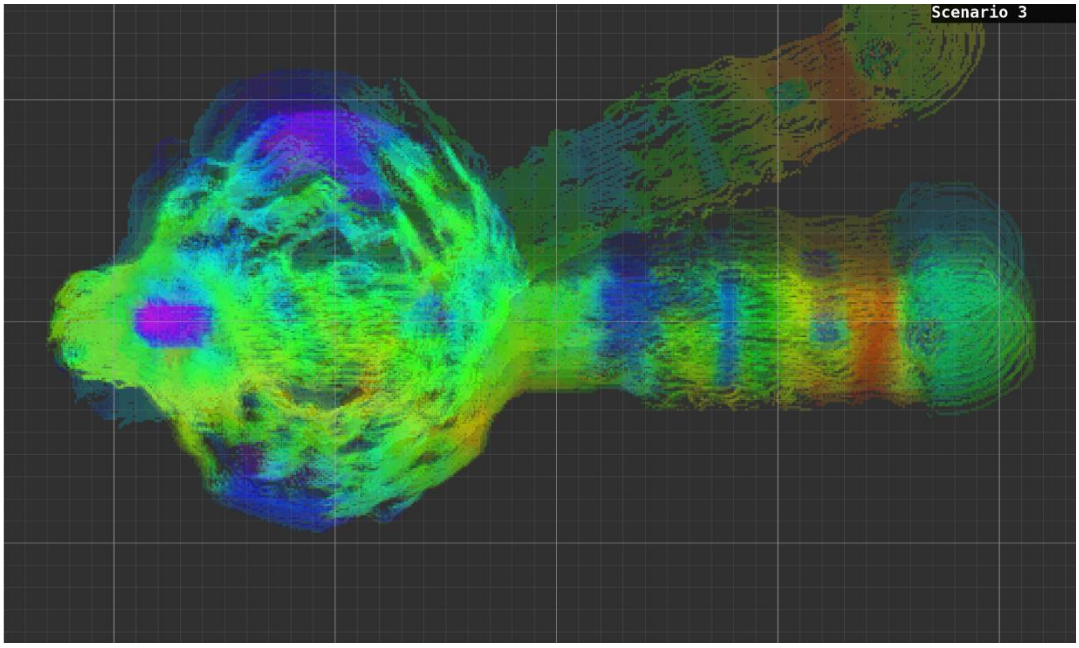

Figure 30. All 10 maps from repeatability tests laid on top of each other.

The worst performing was Run 1, where localisation error can be over 1 metre. Map resulting from Run 1 can be seen on Figure 30, as the one sticking out from the rest of the group. When comparing Run 1 localisation estimate to the one provided by odometry for scenario 3 on Figure 29, it can be seen, that the mean and median localisation error are slightly smaller than odometry, but the maximum localisation error can be higher.

To better analyse the result from Run 1, the pose graphs during different phases of mapping can be seen on Figure 31. The sensor initially completed two small loops with a good location estimate. Then, when returning to starting position it seems to make an incorrect loop closure before leaving to perform movements from scenario 1.

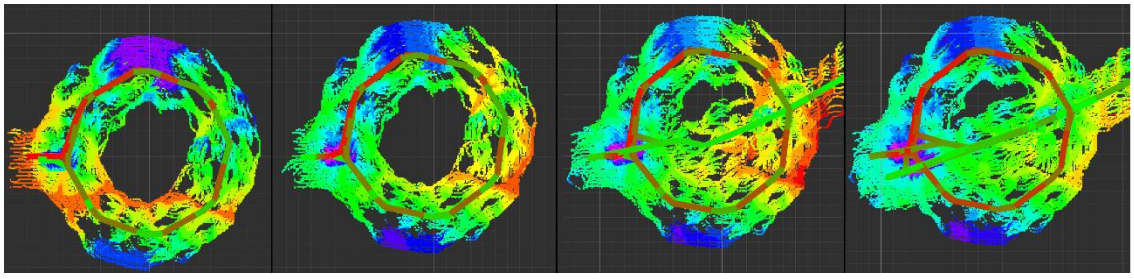

Figure 31. Map with Pose graphs from scenario 3, run 1 of the repeatability tests.

4.2 **Future work**

This thesis provides a proof of concept for a SLAM algorithm that relies on unconventional sensing modality. However, the field is very novel, and many avenues for

future work are possible, from sensor design and SLAM algorithm improvements to software optimisations.

The parameter tuning was performed by visual inspection during this thesis, it is likely that some improvements can be made. It would be interesting to see, how different parameters affect the scan matching quality and if the results presented here can be improved, by adjusting the parameters to more optimal values for this sensor configuration.

To validate the results from the simulator, the sensor grid should be tested in real-world conditions. Experiments similar to the ones presented in this thesis should be performed with the physical prototype sensor grid and analysed.

Another possible improvement would be to implement scan matching odometry back into the SLAM pipeline. A well performing scan matching odometry should decrease odometry errors when discovering unmapped regions. This could improve the localisation estimate during operation, and by having a better localisation estimate it would also increase the loop lengths over which closures can be performed. It could also decrease odometry error in short term and allow for larger sections to be combined into a single keyframe, as described in section 3.4.4.

An optimization can be made, by more carefully selecting the frames, that are added to the pose graph map. Currently all measurements performed by the whisker sensor are integrated into a composite scan and eventually added to the pose graph. It may be possible to decrease the number of scans added to the composite scan and also the number of composite scans added to the pose graph map, by defining some heuristics. For example, if the map already has a large section of the current keyframe covered by the map, then it could just perform localisation and discard the current keyframe. Similarly, it may not be necessary to add all the points generated from sensor measurements the keyframe.

The current sensor implementation in simulation opens up another possibility into researching the sensor design. It is possible to modify the sensor in a way, that allows to measure exact contact points for each whisker in the sensor array. This could allow to determine, whether the tip-contact assumption described in 3.1 actually introduces a large error, or if the effect is negligible. Using the simulator and provided framework it would

be relatively cost effective to investigate, whether the sensor could benefit from such a modification.

The whisker sensor grid is currently measuring only the floor of the terrain. Therefore, it mostly measures varying height contact points, creating a height map of the ground. One could argue that the z-axis coordinate of a measurement has a higher significance than the x or y coordinate of a measured point. For a relatively flat terrain, the accuracy of the whisker sensor could be higher in z-axis than in x or y axis. Therefore, it could be beneficial to see whether having the z-axis value measurements scaled higher than x and y would increase scan matching accuracy. More generally, whether weighing the values in a dimension perpendicular to the whisker sensor array base more, would increase scan matching accuracy during loop closures. This could be tested by multiplying the z-axis coordinates of points by a scalar during loop filtering and performing SLAM with the modified data. The resulting map would no longer be metric, but in our simplified case, it could be turned metric, by dividing the z-axis coordinates by the same scalar, that they were multiplied with.

A possible improvement to scan loop closures could be made by matching multiple scans at once. The current implementation tries to match new sensor measurements with previous measurements in isolation. This works well for lidar-generated point clouds however, the whisker sensor has extremely short range and limited field of view compared to a modern lidar scanner. This can introduce a situation, when descriptive features in a new keyframe point cloud may be split between different scans in the graph-SLAM map keyframes. The current implementation only chooses the best match from existing scans. Increasing the area that can be matched may increase the localisation accuracy and lead to more accurate loop-closure constraints in the pose graph.

5 Summary

This thesis set out to create a proof-of-concept for a SLAM algorithm based on a tactile whisker sensor grid. For the purposes of achieving this goal, a state-of-the-art lidar SLAM algorithm was modified to work with the whisker sensor grid measurement data. To evaluate the performance of the SLAM algorithm, a simulation framework for the sensor grid was developed and test scenarios were run in simulation.

To validate the resulting SLAM framework, five different scenarios were run, with each of these covering a different set of movements. In addition, the repeatability of the results was tested, by performing 10 reruns of SLAM on the same scenario.

Based on the simulated test scenarios, the SLAM algorithm performed better than odometry in vehicle position estimation, suggesting improved localisation capabilities. The map representations produced by the algorithm were visually similar to the ground truth.

The results from experiments were not identical in each run, but repeatable. These results provide a proof-of-concept, that a whisker sensor grid can be used for mapping and localisation purposes, while highlighting limitations and possible improvements. To further evaluate the concept, real-world testing is necessary.

References

[1] L. Lopes *et al.*, "ROBOMINERS – Developing a bio-inspired modular robot-miner for difficult to access mineral deposits," *Adv. Geosci.*, vol. 54, pp. 99–108, Oct. 2020, doi: 10.5194/adgeo-54-99-2020.

[2] A. Ristolainen, J. A. Tuhtan, A. Kuusik, and M. Kruusmaa, "Hydromast: A bioinspired flow sensor with accelerometers," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9793, pp. 510–517, doi: 10.1007/978-3-319-42417-0_55.

[3] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, Dec. 2010, doi: 10.1109/MITS.2010.939925.

[4] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Visual SLAM: Why filter?," *Image Vis. Comput.*, vol. 30, no. 2, pp. 65–77, Feb. 2012, doi: 10.1016/j.imavis.2012.02.009.

[5] F. Dellaert, "Factor Graphs and GTSAM: A Hands-on Introduction," Georgia Institute of Technology, 2012. Accessed: May 10, 2021. [Online]. Available: http://tinyurl.com/gtsam.

[6] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *J. F. Robot.*, vol. 36, no. 2, pp. 416–446, 2019, doi: 10.1002/rob.21831.

[7] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Auton. Robots*, vol. 41, no. 2, pp. 401–416, 2017, doi: 10.1007/s10514-016-9548-2.

[8] R. Dube, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "SegMatch: Segment based place recognition in 3D point clouds," in *Proceedings - IEEE International Conference on Robotics and Automation*, Jul. 2017, pp. 5266–5272, doi: 10.1109/ICRA.2017.7989618.

[9] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," vol. 34, pp. 189–206, 2013, doi: 10.1007/s10514-012-9321-0.

[10] A. Jacobson, F. Zeng, D. Smith, N. Boswell, T. Peynot, and M. Milford, "What localizes beneath: A metric multisensor localization and mapping system for autonomous underground mining vehicles," *J. F. Robot.*, vol. 38, no. 1, pp. 5–27, Jan. 2021, doi: 10.1002/rob.21978.

[11] M. J. Pearson, C. Fox, J. C. Sullivan, T. J. Prescott, T. Pipe, and B. Mitchinson, "Simultaneous localisation and mapping on a multi-degree of freedom biomimetic whiskered robot," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013, pp. 586–592, doi: 10.1109/ICRA.2013.6630633.

[12] M. Salman and M. J. Pearson, "Whisker-RatSLAM Applied to 6D Object Identification and Spatial Localisation," 2018.

[13] M. Quigley *et al.*, "ROS: an open-source Robot Operating System," *ICRA Work.*

open source Softw., vol. 3, no. 5.2, p. 5, 2009, Accessed: May 11, 2021. [Online]. Available: http://stair.stanford.edu.

[14] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 1321–1326, doi: 10.1109/IROS.2013.6696520.

[15] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004, vol. 3, pp. 2149–2154, doi: 10.1109/iros.2004.1389727.

[16] O. Michel, "Cyberbotics Ltd. webots™: Professional mobile robot simulation," *Int. J. Adv. Robot. Syst.*, vol. 1, no. 1, pp. 39–42, Mar. 2004, doi: 10.5772/5618.

[17] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, Mar. 2019, doi: 10.1177/1729881419841532.

[18] S. Chitta *et al.*, "ros_control: A generic and simple control framework for ROS," *J. Open Source Softw.*, vol. 2, no. 20, p. 456, Dec. 2017, doi: 10.21105/joss.00456.

[19] "gazebo_ros_pkgs/gazebo_ros_control at noetic-devel · ros-simulation/gazebo_ros_pkgs." https://github.com/ros-simulation/gazebo_ros_pkgs/tree/noetic-devel/gazebo_ros_control (accessed May 05, 2021).

[20] "hector_gazebo/hector_gazebo_plugins/src at melodic-devel · tu-darmstadt-ros-pkg/hector_gazebo." https://github.com/tu-darmstadt-ros-pkg/hector_gazebo/tree/melodic-devel/hector_gazebo_plugins/src (accessed May 02, 2021).

[21] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 1–4, 2011, doi: 10.1109/ICRA.2011.5980567.

[22] "koide3/hdl_graph_slam: 3D LIDAR-based Graph SLAM." https://github.com/koide3/hdl_graph_slam (accessed May 03, 2021).

[23] M. Magnusson, "The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection," Örebro universitet, 2009.

[24] A. V Segal, D. Haehnel, and S. Thrun, "Generalized-ICP (probailistic ICP tutorial)," *Robot. Sci. Syst.*, vol. 2, p. 435, 2009.

[25] K. Koide *et al.*, "EasyChair Preprint Voxelized GICP for Fast and Accurate 3D Point Cloud Registration Voxelized GICP for Fast and Accurate 3D Point Cloud Registration," 2020.

[26] "SMRT-AIST/fast_gicp: A collection of GICP-based fast point cloud registration algorithms." https://github.com/SMRT-AIST/fast_gicp (accessed May 13, 2021).

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Andreas Nagel

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Blind Mapping and Localisation for Small-Scale Mining Robots”, supervised by Asko Ristolainen
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

17.05.2021

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.