Bachelor Project

Czech Technical University in Prague

F3 Faculty of Electrical Engineering
Department of Cybernetics

Sensor Fusion for Mobile Robot Localization

Václav Plavec

Supervisor: Ing. Vladimír Smutný, PhD.
Subfield: Cybernetics and Robotics
January 2021

Acknowledgements

I would like to thank the CTU for being such an excellent *alma mater* for me. I would also like to express my thanks to my supervisor of this project, Ing. Vladimír Smutný, PhD., not only for his valuable advice, but also for his enormous patience he had with me during by work on this project. Last but not least, I would like to express my thanks and gratefulness to my girlfriend, who held my hand during both the most joyful and the most difficult times, which were caused by the pandemic during the time I was working on this project.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 5 January 2021

Abstract

This work is dedicated to studying of uncertainty of the estimation of a pose of the mobile robot Clearpath Robotics Jackal, based on information from various sensors. Not only the estimation of a pose, yet also the uncertainty of this estimation are impotant not only as a primary source of the information about the robot's pose, but primarily for application in various more complex algorithms for localization of a robot and for mapping of the surrounding space.

The goal of this work is modeling of the uncertainty of the estimation of the pose, based on experimantal results, and its approximation, depending on velocities of the robot and on the driven distance. Partial goals are also the study of characteristics of data of the particular sensors, which provide information about the pose of the robot, such as rotary encoders on the motors, which provide information about their rotation, or a gyroscope and an accelerometer of an inertial measurement unit (IMU).

The output of the work is an estimated approximated model of the uncertainty of the estimation of the pose, obtained by fusing information from the rotary encoders mounted on the motors of the robot and of the IMU, which is used by the robot. This model is finally expressed so, that it is possible to compose it into the algorithm for simultaneous localisation and mapping (SLAM), which is actually runs on the robot and which uses the estimation of the pose for its initialization, yet it should use also the uncertainty of the estimation, which it does not do. This algorithm localizes the robot based on the data from a LiDAR and simultaneously creates a map of its surrounding space.

Keywords: robot, uncertainty, pose, odometry, EKF, Leica, HTC Vive, sensor, normal distribution, error, covariance, SLAM, ROS, IMU, rotary encoder

Supervisor: Ing. Vladimír Smutný, PhD.
Department of Robotics and Machine Perception,
Czech Institute of Informatics, Robotics and Cybernetics,
Jugoslávských partyzánů 1580/3,
Prague 6

Abstrakt

Tato práce se věnuje studiu nejistoty odhadu polohy mobilního robotu Clearpath Robotics Jackal na základě informací z několika senzorů. Nejen odhad polohy, ale i nejistota tohoto odhadu jsou důležité nejen jako primární zdroj informace o poloze robotu, ale především pro aplikaci v různých komplexnejších algorimtmech pro lokalizaci robotu a mapování okolního prostoru.

Cílem této práce je modelování nejistoty odhadu polohy na základě experimentálních výsledků a její aproximace v závislosti na rychlostech robota a ujeté vzdálenosti. Dílčími cíli jsou rovněž studium charakteristik dat jednotlivých senzorů, poskytujících informaci o poloze robotu, jako jsou rotační enkodéry na motorech, které poskytují informaci o jejich otáčení, či gyroskop a akcelerometr v rámci inerciální měřicí jednotky (IMU).

Výstupem práce je odhadnutý aproximovaný model nejistoty odhadu polohy získaného sloučením informací z rotačních enkodérů na motorech robota a z IMU, kterou robot užívá. Tento model je nakonec vyjádřen tak, aby bylo možné jej později zakomponovat v algoritmu pro simultánní lokalizaci a mapování (SLAM), který aktuálně na robotu běží a který užívá odhad polohy pro svou inicializaci, avšak měl by uvažovat také nejistotu tohoto odhadu, což nedělá. Tento algoritmus na základě dat z LiDARu lokalizuje robota a současně vytváří mapu okolního prostoru.

Klíčová slova: robot, nejistota, poloha, odometrie, EKF, Leica, HTC Vive, senzor, normální rozdělení, chyba, kovariance, SLAM, ROS, IMU, rotační enkodér

Překlad názvu: Slučování informací z více senzorů

Contents

1 Introduction **1**

2 State of the Art **3**

2.1 Probabilistic methods and uncertainty representation 3

2.2 Estimating odometry from dead reckoning using only proprioceptive sensors or a compass 4

2.3 Simultaneous localization and mapping (SLAM) and combining data from exteroceptive sensors with data from proprioceptive sensors . . . 4

2.4 Multi-robot coordination and cooperation 7

2.5 Neural networks and machine learning in the process of robot localization 7

2.6 Types of wheeled unmanned ground vehicles (UGV's) in terms of their movement 7

2.6.1 Differential drive kinematic model 8

2.6.2 Skid-steering model 8

2.6.3 Ackermann steering kinematic model 9

2.7 More accurate model of the skid-steering kinematics 10

2.8 The base of this work 11

3 Used Hardware and Software **13**

3.1 The Clearpath Robotics Jackal UGV setup 15

3.1.1 The hardware 15

3.1.2 The software 17

3.1.3 Data sources providing information about the pose and velocities of the robot 18

3.2 HW and SW equipment for the experiments 19

3.2.1 HTC Vive tracker 20

3.2.2 Leica AT403 absolute tracker 20

3.2.3 Measuring the robot's pose with the laser tracker guided by the tracking system 21

4 Theory **25**

4.1 Extended Kalman Filter and the EKF-fused odometry 25

4.2 The robot's kinematic model and its use 27

4.2.1 The Skid-Steering kinematic model 27

4.2.2 The controller's mathematical background 31

4.3 Boxan's extension of Nováček's NDT SLAM 32

4.4 Finding a transform between 2 sets of corresponding points 34

4.5 Estimating poses of the retroreflectors for guiding the laser tracker using the tracking system . 36

4.6 Modeling the uncertainty of the robot's EKF-filtered odometry 40

4.6.1 Modeling of the mean error vector and covariance matrix from the experimental data 40

4.6.2 Approximation of the contribution to the mean error vector and to the covariance matrix depending on the velocities and the increment of driven distance 42

5 Implementation **45**

5.1 The control of the robot's movement 45

5.1.1 The EKF-fused odometry . . . 45

5.1.2 Jackal's driving controller and its optional setup parameters . . . 45

5.2 Libraries used in the scripts 50

5.3 The script for controlling the robot's driving and generally its measurement 51

5.3.1 Command line arguments of the script 51

5.3.2 Commands of the gamepad . . 53

5.4 The script for evaluating the measured data 54

5.4.1 Evaluating the results of the 2nd experiment 54

5.4.2 Plotting the data from the odometry and IMU topics for a repeated movement during the 3rd experiment 54

5.4.3 Extraction of the data from baglists (lists of rosbags) 54

5.4.4 Computing and approximating the resulting mean error vectors and covariance matrices from the extracted data 55

5.4.5 Replotting the saved data of odometries from the topics 56

6 Experiments **57**

6.1 The reference coordinate frames considered in the experiments 57

6.2 Observing the characteristics of the proprioceptive sensors of the robot 59

6.2.1 Results of the experiment . . . 62

6.2.2 Implications 65

6.3 Measuring the accuracy of the tracking system data 71

6.3.1 Performing the experiment . . 71

6.3.2 Results of the experiment . . . 72

6.4 Measuring and estimating the uncertainty of the EKF-fused odometry of the robot 77

6.4.1 Performing the experiment . . 77

6.4.2 Processing the data 78

6.4.3 Results of the partial experiments for the different commanded velocities 79

6.4.4 Results of the whole experiment 138

7 Suggestions for the Further Use of the Results 143

8 Conclusion 145

Bibliography 147

A Structure of the CD 151

B Project Specification 153

Chapter 1

Introduction

The **odometry** is the fundamental source of information about a robot's pose. There exist many methods enabling a robot to determine or rather estimate its pose, yet a significant amount of them does not perform their proper update cycle with sufficiently high frequency. The **encoder odometry** is computed from the information about the rotation of robot's motors and thus its wheels, taken from rotary encoders attached to the motors. It is then employed to estimate the robot's pose with sufficiently high frequency, while using more advanced and complex methods, between the 2 subsequent proper update cycles of the particular method's algorithm performing. It is usually due to the fact that such an advanced algorithm is computationally and time demanding. In order to make the encoder odometry more accurate, its data can be fused with data of another sensor, providing similar kind of information about the robot's pose, or velocities or accelerations, with comparably high frequency. One example of such a sensor is an **inertial measurement unit** ("**IMU**"), which may provide information about the robot's accelerations and its angular velocities. The most widely used method of fusing such 2 or more sources of information about robot's pose is the **Extended Kalman filter** ("**EKF**"), providing the **EKF-fused odometry**, which considers not only the information about pose from the sensor, yet also the uncertainty of the information.

Another case when the use of the encoder odometry is necessary is **when localization methods maintained by other types of sensors are not accurate enough**, especially under certain conditions. One example of such combination of a sensor method and adverse conditions is using **visual localization** (using a camera and computer vision methods) in a **long visually monotonous corridor**. It is by the way the case of corridors in the B building of the CIIRC (Czech Institute of Informatics, Robotics and Cybernetics). Next example directly related to this work is performing Simultaneous Localization and Mapping (**SLAM**) using **LiDAR** scan in a **long narrow corridor**, as the LiDAR data tend to curve the perceived corridor with increasing angle between the LiDAR ray and the wall surface, which is again related to the corridors in the B building of the CIIRC.

The main goal of this work is to estimate a probabilistic model of the **uncertainty of the EKF-fused odometry** of the **Clearpath Robotics Jackal Unmanned Ground Vehicle** (“**the robot**”). The particular goals are to estimate it, to model it and to **approximate its dependence** on the commanded linear and angular velocity of the robot, as well as on the driven distance. It shall be approximated and expressed in such a way that it can be **integrated into the SLAM algorithm**, which is currently running on the robot and currently does not use any probabilistic model of the uncertainty of the EKF-fused odometry.

Chapter 2

State of the Art

2.1 Probabilistic methods and uncertainty representation

The **2-dimensional normal distribution** is used for the representation of a mobile robot's position estimation uncertainty. Durant-Whyte and Leonard [10] studied the uncertainty of mobile robot's position acquisition from an exteroceptive sensor (**sonars** and **infra-red sensors**) and visualized it as gradually evolving ellipsis of confidence along with a method of correcting it after obtaining information about the position from a beacon using the extended Kalman filter (**EKF**). This work provided a versatile tool and description of how the **EKF uncertainty** evolves in terms of its **geometrical representation**. Watanabe and Yuta [35] built on this work, using also the the angles of the rotation of motors read by a **rotary encoder** attached to the motors assuming a 1:1 gear ratio used to determine the robot's planar pose and driven distance (**"encoder odometry"**) for **dead reckoning** in terms of their required speeds. They modeled the **uncertainty** of a mobile robot's encoder odometry used for the dead reckoning and expressed the **error ellipsoid** visualizing the uncertainty directly in the analytic matrix form from the total covariance matrix. They also used **lighthouses** as means of **correction** of the robot's **pose estimation uncertainty**. Komoriya *et al.* [18] extended that work and as means of the **pose estimation uncertainty correction** they used **point and line type landmarks**. Choi, Lee and Lee [5] continued with this work and based on it they designed and realized a method of such **landmark-corrected** mobile robot's position estimation using **RFID landmarks** besides the **encoder odometry**. They also made the **uncertainty ellipsoid** computing more effective by applying the **SVD of the covariance matrix**.

Julier [17] suggested several algorithms for the **distributed fusion of Gaussian Mixture Models** using the **Chernoff Information**. He used the **Bayes rule** and the **Chernoff**

information and provided a versatile tool for **combining the Gaussians**.

2.2 Estimating odometry from dead reckoning using only proprioceptive sensors or a compass

What concerns more recent work in the field of fusing information from more types of sensors, Hoang *et al.* [14] used the **EKF** for estimating the mobile 2-wheeled robot's position from the the **encoder odometry**, a **compass**, a **laser range finder** (**"LiDAR"**) and an **omni-directional camera**. This work provided a detailed depiction of how the trajectory estimation of each single sensor and of their EKF-fused information evolve through the increasing time of the robot's movement and how they differ from the true path. They also provided a detailed depiction of principles of each sensor's function as well as the idea how to work with the covariances not only along with the **EKF**. Recently, Cheng *et al.* [8] proposed an advanced mobile robot self localization algorithm, **Limited memory Kalman filter (LMKF) with exponential fading factor**, which uses the exponentially fading weights of recent sensor measurements in order to improve the localization errors. They use it to fuse data from **wheel odometer**, a 3-axis digital **gyroscope**, a 3-axis **acceleration sensor** and a 3-dimensional magnetoresistance electronic **compass**. They provide a detailed mathematical description of the algorithm and show that it is capable of **filtering out random compass errors** and **reducing the cumulative errors of the gyroscope and the odometer**.

2.3 Simultaneous localization and mapping (SLAM) and combining data from exteroceptive sensors with data from proprioceptive sensors

Concerning the recent proceedings in the **Simultaneous Localization and Mapping** (**SLAM**), Cai and Zhong [7] proposed a **SLAM** algorithm based on the **Adaptive square root cubature Kalman filter**. In their work they provided a detailed description of their algorithm and they showed that it has quite decent performance. Lee *et al.* [21] examine the observability of conventional **SLAM** by using the **Fischer Information Matrix** and improve its **observability** by proposing their **Dual-sensor-based Vector-field SLAM** (**DV-SLAM**) along with a special (so called **Rao-Blackwellized**) **Particle filter**, which is fully observable, applied on **compass/magnetometer** sensors. They analyze its performance on a platform of a robotic vacuum cleaner with 1 or 2 magnetometers mounted on it in various angles towards each other, data of which are fused along with the estimated odometry from wheels control input, and found out that a setup of 2 magnetometers placed in angle of π rad towards each other is fully observable and the best solution among all possible placements. Będkowski *et al.* [6] propose an open source robotic **3D mapping framework** with a vast library of tools to work with **point clouds**, their analysis, importing, exporting etc., distributed along with a bunch of datasets. It has been tested on **Clearpath Husky** mobile robot. They depict how the **odometry and SLAM uncertainties** evolve through time in a detailed way unless they are properly processed. Bahreinian *et al.* [12] examine the performance of **RMF-SLAM** and **AMF-SLAM** dealing with **relative and absolute landmark positions** respectively. They found out that both the methods reach significantly better results that just pure **EKF**. Yuan *et al.* [37] deal with the **RGB-D sensor based Visual SLAM** for localizing and navigating a restaurant serving robot. They use the Microsoft Kinect (**RGB-D camera**) for **landmark detection**, SICK (**laser scanner**) for the map matching and the **wheels odometry** data and combine it all in the **EKF**, which they proved to be quite an effective combination. Similarly, Fan *et al.* [11] use a **laser range sensor** for metric mapping, a **barometric pressure sensor** for detecting floor transition and the Microsoft Kinect sensor (**RGB-D camera**) for collecting **3D environment information**. They use the **Monte Carlo localization** (a **particle filter** localization) for the localization based on the **laser range sensor**. They showed that such combination is indeed convenient for this task. In their work they also describe how they use and combine the data in the **EKF**.

One of the most suitable works to proceed from in this work is the work of Zhou *et al.* [39], in which they use a **LiDAR** odometry along with **NDT** based scan matching for outdoor mobile robots localization in environments where the GPS localization is not available, as well as with the **3D kinematics** model based on **encoder** and **IMU** (Inertial Measuerment Unit - accelerometer and gyroscope) data used for the **dead reckoning**. They conveniently describe how the **NDT** (Normal Distribution Transform) algorithm works and what it actually does. The main contribution for this work is their description of how they fuse all the data together in the **EKF**. They use the **LiDAR/NDT** odometry at low frequency to periodically correct the dead reckoning accumulated uncertainty/error. They showed that when using recent 20 frames to match the position is efficient comparably with using all frames to match the map and that this combination of sensors is more than suitable for solving such localization problem. Another contributive work is done by Akai *et al.* [3], where they implement accurate localization of an autonomous car by fusing data from **3D LiDAR** matched by **3D NDT** after having split the map into voxels along with the resulting **Hessian matrix** computed during the optimization process and **dead reckoning** based on the **encoder odometry** and on **IMU** data (there were also data from other sensors, but they didn't use it) with the EKF. In the work they visualize the uncertainty with the **uncertainty ellipsoid**. Their results include formulated facts about that method, including the idea that robust localization could be achieved by **fusing the NDT and dead reckoning results with EKF even if the pose estimated by NDT was disturbed**. Also Uyulan *et al.* [33] made significant contribution to this field, fusing information from **dead reckoning** based on the 4-wheeled mobile robot **encoder odometry** with localization based on **LiDAR** scan matching with an **occupancy grid map**. They treat the **position error accumulation** by comparing performance of **Extended Kalman Filter** (**EKF**), **Unscented Kalman Filter** (**UKF**), **Unscented Information Filter** (**UIF**) and **Extended Information Filter** (**EIF**). They provide a detailed depiction of how the **error ellipsoid** evolves through time and after a correction from an accurate measurement as well as a detailed description of how each of the examined algorithms work. They concluded with the discovery that the **UIF** performs with **better accuracy** and the **EIF** performs with **better stability** than the **Kalman filters**. Returning to the autonomous vehicles, Li *et al.* [22] propose a **rigid point set registration** based on **Cubature Kalman filter** (**CKF**), which they describe in a very detailed way and apply it in the process of localization of an autonomous car based on fusion of **3D LiDAR**, **GPS**, **IMU** and **wheel encoder** data. They show that it **approximates the nonlinearities better** than the ordinary EKF as well as it is **more robust to noise, initialization misalignment and outliers** along with their proposed **correspondence between 2 point sets based on local feature vector** in comparison with another methods like the **ICP** (**Iterative closest point**) or the **NDT**.

As another promising proceeding related to problems being treated in this work, Yu and Zhang [36] propose an **Improved Hector SLAM Algorithm** based on **Information Fusion** for Mobile Robot, fusing **wheel encoder** and **IMU** odometry along with the **LiDAR SLAM**. As a main problem, they studied the problem of **SLAM** from **degenerated LiDAR data in a long corridor**, which is one of the main problem this work is trying to deal with, and showed that its performance is significantly **better** than the **conventional SLAM**. A very similar problem has been studied by Laconte *et al.* [20] who focused on the problems of **LiDAR** localization in a **long narrow tunnel**, yet in a context of **3D mapping**. They first show that the bias causing a **curvature of a LiDAR-measured corridor** depends on where in terms of the corridor width the robot is situated, whether in the middle or closer to either wall. They examined the problem and discovered that the **distance** measured by the **LiDAR** strongly **depends on the angle between the ray and the wall**. They studied the cause of this phenomenon and finally came with a solution, modelling the laser ray as a **Gaussian light beam**, which explains why the LiDAR senses the point on the wall to be **closer than really is**. In fact, the error is up to 20 cm at a distance less than 10 m, so they propose a **method of removing the bias** of points of surfaces scanned under larger incidence angle.

Toroslu and Doğan [32] are among others who deal with finding an effective algorithm for a mobile robot **SLAM** fusing data from an **ultrasonic sensor** with data from **IMU** and **wheel encoder** written in Python for a platform based on Raspberry Pi. They show that the **IMU** data are **affected by signal noise and wibrations from wheels** and they filter the **encoder** and **IMU** data using a **complementary filter** in order to remove noise and errors. Liu *et al.* [24] propose the **Unscented Kalman filter** (**UKF**), **Augmented Monte Carlo localization** (**AMCL**) and **2D Distribution-to-Distribution** (**D2D**) **NDT** for improving the accuracy of fusion of **2D LiDAR**, **IMU** and **encoder odometry** of a mobile robot as well as of the **SLAM**. They experiment with the Clearpath Husky robot and provide a detailed description of their algorithm combining all the mentioned methods and show that it reaches very high accuracy in comparison with the sole methods.

2.4 Multi-robot coordination and cooperation

Proceedings have been also made in the field of **multi-robot coordination and SLAM**. Lin *et al.* [23] propose a hybrid positioning method for **multi-robot SLAM** allowing to simultaneously coordinate multiple robots as well as to let them simultaneously create one **common map** based on the **LiDAR NDT**-matched measurement using **visual markers** for their cooperative identification, producing decent results. A detailed description of obtaining and computing **sensor data variances** for the sensor information fusion is proposed by Inoue *et al.* [15]. They study swarm robot systems consisting of large numbers of cooperative simple physical robots, which have only the exteroceptive information about **distances towards one another** and the **velocity control input**, and propose an algorithm for their position estimation that they show to be more accurate than fusion of encoder odometry and exteroceptive sensors.

2.5 Neural networks and machine learning in the process of robot localization

Another branch of research recent proceedings concerns learning of mobile robots with or without the use of **neural networks** either in decision or in sensor information processing in terms of both camera or lidar data processing and sensor information fusion. Ostafew *et al.* [28] experimented with the **Clearpath Husky** robot in rough and various outdoor terrain. They implemented a control algorithm that uses **visual odometry** for the localization and estimates its states and uncertainties based on an apriori **Gauss-probability-distributed model** along with **learned model**, which the robot learns while moving. They also model a $\pm 3\sigma$ boundaries representing the **uncertainty** of its position along with its mean based on the **Monte Carlo localization**. Cho *et al.* [9] for example apply a **three-dimensional Convolutional neural network** (**3D CNN**) for estimating robot's **six-degrees-of-freedom** (**6DOF**) **odometry** from sparse **3D LiDAR** data using **deep learning**.

2.6 Types of wheeled unmanned ground vehicles (UGV's) in terms of their movement

In the area of wheeled unmanned ground vehicles (**UGV**), there exist many different kinds and models. They can be generally divided into the following 3 groups of **kinematic models** models.

2.6.1 Differential drive kinematic model

This kinematic model of UGV's has usually, but not necessarily a round shape of its base. Such UGV's feature

- **A set of 2 independent wheels** aligned in **one axis**, typically situated along the minor symmetry axis (in case of rectangular robot) or along the robot's diameter (in case of a robot with a circle base
- **2 possible options of third support point:**
  1. A **glider**, which means typically a round object with smooth rounded surface gliding on the wall
     - This is very beneficial compared to the second option below, because apart from the fact that it means friction, it does not significantly affect the movement of the robot i.e. during steering
  2. A **trailing wheel**, which is the kind of a rotary wheel that can be found i.e. on a shopping cart
     - Although it results in much less friction during a straightforward movement, it significantly affects the robot movement direction during its steering, because it is typically in such form that the rolling axis of the wheel does not intersect the turning axis

This model of UGV is the easiest for modeling its steering, because (apart from the case of sharp change of the movement direction in the case of a base with a trailing wheel) it is only about moving 2 wheels around circles with 2 different peripheral speeds. For this ease of the movement modeling, it is often used in basic robotic courses,

2.6.2 Skid-steering model

This kinematic model of an UGV is more complex that the TurtleBot model and is represented by the robot studied in this work, the Jackal UGV, which will be described later in the subsection 3.1. Its complexity is not only caused by its more complex construction, but also by the more complex modeling of its movement, especially the steering. This model's base features typically 2 independent pairs of linked wheels, one linked pair on each side. The complexity of the steering of this model is in the fact that in order to steer, the wheels have to skid.

That leads to significant nonlineary of the movement model, which is described in detail below in the subsection 4.2. This nonlinearity means a problem for proper localization of the robot only from the encoder odometry, because the IRC's of the motors can sense only the rotation of the wheels and thus their forward speed, but not their skidding and thus their lateral speed. Yet the lateral speed of the wheels makes a significant contribution to the movement of the robot and thus has to be considered.

There are 2 mainly used methods of considering the skidding and the consequent lateral speed of the robot's wheels in the self localization.

The first method, which is used in the majority of cases, is to perform a method of sensor data fusion from the encoder odometry and another device such as IMU, which gives more accurate information about the robot's direction and its change. This method has been described and analyzed among others by Wang *et al.* [34], whose work is studied below in the subsection 4.2. Their article provides an **overview of the mathematical background for this model**, which will be used later in this work.

The second method has been proposed among others by Kozłowski and Pazderski [19], by Wang *et al.* [34] or by Rabiee and Biswas [29], who studied the skid-steer kinematic model particularly of the Clearpath Robotics Jackal UGV and created quite an accurate model of the Jackal's movement, considering actually the friction of the wheels and their skidding, which is mentioned below in the subsection 2.7.

2.6.3 Ackermann steering kinematic model

This kinematic model represents the model that is mainly found in a car. It features a pair of diferentially driven rear wheels on a common axis and a special construction of the support for the pair of front wheels. The specialty about the construction of the support for the front wheels is about the geometry that allows the car to steer so that the **axis of the rear wheels and both the axes of each of the front wheels intersect in one point** representing the actual **center of the rotation**. This mechanism of the construction is referred to as the **Ackermann steering geometry**. Its scheme can be seen in the figure 2.1. Zhao *et al.* [38] suggested, among others, a decent method of design of such model.

This model requires a more complex construction of the steering mechanism, yet the steering itself can be modeled very easily. In addition, during steering of this model, almost zero friction is created in comparison with the skid-steering model.

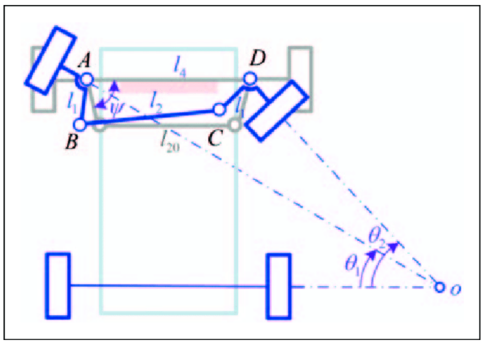


Figure 2.1: Ackermann turning geometry scheme in car-like model [38]

2.7 More accurate model of the skid-steering kinematics

In terms of the the **friction-based dynamical and kinematical** model of a **skid-steering** wheeled mobile robots, Kozłowski and Pazderski [19] studied the problem of dynamic modeling and proposed a driving algorithm considering the **friction** and **lateral velocities** of the robot's wheels during skidding. Their algorithm considers kinematics and dynamics of a 4-wheeled skid-steering mobile robot and to solve the kinematic problem and maintain stabilization it uses the idea of a **kinematic oscillator** for the regulation.

Also Wang *et al.* studied the **skid-steering** kinematic model first from the view of an approximation as an ideal differentially driven robot, which has been mentioned. They also studied the **dynamics** of such model and proposed a model considering the robot's **dynamics** in terms of **friction** and **rolling resistance**, which is able to achieve relatively accurate results **without using any additional sensors**.

Later, Rabiee and Biswas [29] studied the **Clearpath Jackal**'s **friction-based kinematics** and proposed a method of its **skidding lateral velocity compensation**, which showed significantly better accuracy in terms of the **dead reckoning odometry** in comparison with the odometry predicted based on the **encoder odometry**.

2.8 The base of this work

One of methods of robot localization, which cannot be performed with sufficient update rate, as has been described in the introduction, is the **NDT SLAM** (Normal Distribution Transform Simultaneous Localization And Mapping) proposed by Nováček [27], who extended and improved Jelínek's work [16]. He uses the **LiDAR** data matched by the **NDT** with the **occupancy grid** created from **CAD** drawing of the map of the 6th floor in the building B of the Czech Institute of Informatics Robotics and Cybernetics (CIIRC) for the localization of the **Clearpath Robotics Jackal Unmanned Groung Vehicle** (**Jackal UGV**), which will be studied in this work. Each update of the algorithm's execution is performed only after having driven certain minimum distance or turned certain minimum angle, because it is so computationally demanding that is has to be run on an external computer. It uses the **data of the encoder odometry fused with data from the robot's IMU** by the **EKF** for the higher-rate estimation of the robot's pose between the 2 subsequent updates of the algorithm.

The problem with his work is the fact that he uses the robot's odometry only for the robot's pose initialization in the **SLAM** and to estimate the robot's pose between the algorithm's 2 subsequent updates, as has been mentioned, yet his algorithm immediately **forgets the odometry as well as its uncertainty** before the following algorithm's initialization and thus has no prior information helping the algorithm to converge to the correct solution. This imperfection manifests in the most significant way when the robot is trying to localize itself in a long narrow corridor, which tends to have repetitive patterns in its relief.

Later, another work about Jackal's localization has been written by Boxan [4], who solved the problem of **respecting visibility** of the map entities during the **NDT SLAM**. He **extended Nováček's work**, particularly his **score function**, which is optimized during the SLAM processing and will be described in the Chapter 4, in the section 4.3 on page 32, by **adding information about the visibility** to it. He also proposed an **approximation of the score function**, which improved the performance of the SLAM algorithm and particularly its results **in long narrow corridors**. Besides it he learned that the CAD drawing of the CIIRC's 6th floor is inaccurate. He therefore corrected the CAD drawing, which **improved the accuracy** of the **NDT SLAM** running on the Jackal UGV. Last but not least, he did not consider the real model of the uncertainty of the EKF-fused odometry of the Jackal UGV, but he prepared his program for the **further integration of the information about the uncertainty of Jackal's odometry**, where he temporarily a null covariance matrix. Therefore he prepared **great background for this work**. He also prepared software for the visualization of the score function and of the behavior of the SLAM optimizing cycle.

Chapter 3

Used Hardware and Software

This work is solving the problem of accurate indoor localization of the **Clearpath Robotics Jackal Unmanned Ground Vehicle** (**UGV**) based on information from both proprioceptive and exteroceptive sensors.

As has been mentioned in subsection 2.8 on page 11, there has been already done an implementation of **NDT-SLAM** by Nováček [27], who extended and improved Jelínek's work [16]. The problem is that the Nováček's implementation uses the robot's **odometry only for the position initialization** in the **SLAM** and then **forgets the odometry as well as its uncertainty.** This defect manifests in the most significant way when the robot is trying to localize itself in a **long narrow corridor**.

It has been also mentioned that Boxan [4] made a significant progress originating from the Nováček's work. He namely **approximated the** Nováček's **score function** and **added information about visibility** to it. His visibility algorithm considers the information about the movement taken from the **odometry** for the **estimation of the most probable next position** before the next NDT SLAM algorithm run, which runs once per 15 centimeters driven. Although he uses the odometry, he tries to eliminate its uncertainty only by tuning a multiplier of the wheel diameter used by the robot's driving controller, which will be further described below in the subsection 5.1.2. He thus **does not use the probabilistic model of the uncertainty**, but he **prepared his code of the SLAM to integrate the information about the odometry uncertainty into** by filling in the actual covariance matrix.

This work will thus try to **measure, model and approximate the uncertainty of the odometry** and its **evolution along the trajectory in time** so, that it could be integrated to the SLAM algorithm.

First the evolution of the odometry uncertainty will be studied and properly probabilistically modeled based on proper measurements and experiments. It will be subsequently rigorously expressed, modelled and then its evaluation's implementation and integration of it to the existing Boxan's program (extending the Nováček's NDT SLAM), which the Jackal UGV already uses for its localization, will be performed.

The accuracy of the improved SLAM algorithm will be then measured. In the conclusion, the results of the measurement will be studied, evaluated and discussed.

3.1 The Clearpath Robotics Jackal UGV setup

3.1.1 The hardware

The used robotic platform is the Clearpath Robotics Jackal Unmanned Ground Vehicle (**"the robot"** from now on) with several additions, which can be seen in the figure 3.1. This robot's design represents the **Skid-steering model** mentioned above. The steering model will be further described and studied below in the subsection 4.2.

The robot contains the following sensors:

1. **motor rotary encoders** - on both the motors, as both the lateral pairs of wheels are driven each by one motor and are connected with a belt, use as a source of data for the **encoder odometry**
2. **IMU** (Inertial Measurement Unit) - A sensor combining a **gyroscope**, an **accelerometer** and a **compass** (which is not used and has not been much tested) - a component originally mounted by Clearpath
3. **GPS** - not used in the indoor navigation
4. **LiDAR** (SiCK TiM561-2050101) - Laser-based 2D LiDAR
5. **omnidirectional camera** (Basler daA2500-14uc with fish-eye lens Sunex PN DSL215) - Colleague Pánek is just in a process of development of visual odometry using this
6. **HTC Vive tracker** - used for experimental measurement, described in a more detailed way below, in the subsection 3.2.1 on page 20

The computational, controlling and communication intelligence of the robot is divided into these major parts:

1. **MCU** (Microcontroller Unit, original internal component) - controls the motors, power etc. and reads data from the sensors (odometry, IMU, LiDAR, ...)
2. **Internal PC** (Aimb 274) - communicates with and controls the MCU, LiDAR, Bluetooth joystick, LED strips controller (which will be described) and runs the lower-level programs, such as ROS (Robot Operating System, will be described) movement controllers and collision avoidance

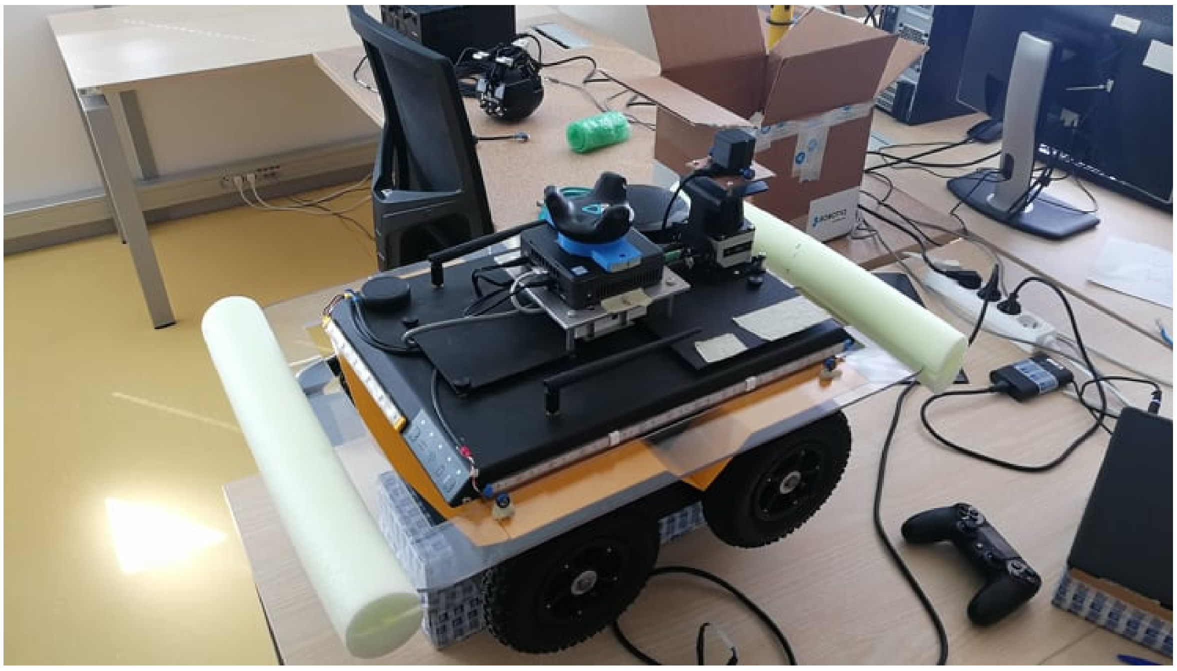


Figure 3.1: Clearpath Robotics Jackal UGV platform

3. **Router** - maintains communication between the internal PC, NUC and possible wired connection
4. **Intel NUC PC** - establishes the wireless access point to communicate with all the PC's, takes image from the camera and runs the higher-level programs such as motion planning, mapping etc.

This mobile robot contains a 28-V 270-Wh Lithium battery, which significantly contributes to the robot's weight of about **16 kg**.

This setup contains also the option to control the robot manually with the **Sony Dualshock 4 V2 Controller**, which is a wireless Bluetooth gamepad and will be referred as **"the gamepad"** in the further text. Besides the manual controlling function, the gamepad also serves as such a "dead man switch", meaning that a user has to keep the **R1** button on the gamepad pressed in order to allow any other software to publish commanded velocities, as well as to be able to control the robot with the gamepad.

3.1.2 The software

The core of all the software running on the robot's PC's is the **Robot Operating System** (**ROS**), which is described below. The internal robot's PC runs the ROS Indigo version and the NUC

The controllers controlling the robot's movement are described below in the subsection 5.1.2.

The Intel **NUC** PC, running the Ubuntu 16.04, runs the **ROS Kinetic** version, whereas the **internal PC** Aimb 274, running the older Ubuntu 14.04, runs the older **ROS Indigo** version.

The controllers driving the robot

The driving of the robot is maintained by the **`diff_drive_controller`** ROS package. This package represents a controller for **driving a skid-steering UGV** and is well parametrizable. It is described in detail below in the subsection 5.1.2 on page 45. This controller is used by the **jackal_velocity_controller** node.

After launching the **bringup_all** launch file, which initializes all the additional hardware including the camera, the LiDAR, the **collision_avoidance** node, maintaining the collision avoidance of the robot when being controlled by the movement planner, launches along with the node maintaining the communication with the gamepad.

Let us now study the robot's ROS topics bearing information related to this work.

3.1.3 Data sources providing information about the pose and velocities of the robot

During the experiments and in the scripts for measurement and for evaluation of the measured data, data from the following ROS topics were taken:

1. `/jackal_velocity_controller/odom` - raw encoder odometry,
2. `/imu/data` - prefiltered IMU data, including
   - The **accelerometer** data **double integrated** for computing the **position** and
   - The **gyroscope** data **integrated** for computing the angle $\boldsymbol{\theta}$, *and*
3. `/jackal_velocity_controller/odom` - odometry as EKF-fused encoder odometry and IMU data.

Another meaningful secondary source about the pose of the robot, used by the SLAM algorithm, is the topic `/scan/filtered`, where the filtered data of the LiDAR are published. Next important topic is the topic `/set_pose`, which enables to reset the pose published by the EKF-fused odometry or to set it to a wanted pose. The topics, where the commanded velocities may be published, are the following:

1. `/cmd_vel_safe` - topic, where the velocities can and should be preferably published from any ROS node running on the hardware contained on and in the robot. These data then go through a security phase, where the following is verified in order to determine whether to let the commanded velocities apply:
   a. whether the robot is not in front of an obstacle (detected by the collision avoidance ROS node according to the actual LiDAR data) *and*
   b. whether the **R2** button on the gamepad is pressed.
2. `/cmd_vel` - the topic where only the commanded velocities gone through the security phase described right above go and should only go.

3.2 HW and SW equipment for the experiments

This work aims to

1. **correctly probabilistically model the uncertainty of the robot's odometry** and to
2. **approximate the dependence of the contribution to the uncertainty** depending on the commanded linear and angular velocity and the distance driven from the previous update of **the Boxan's extension** [4] **of Nováček's NDT SLAM algorithm** [27]
3. **express the approximation** so that it can be used to implement an improved SLAM algorithm by integrating the approximation to Boxan's version of the SLAM algorithm

During the experiments we will record the evolution of the robot's **EKF-filtered odometry** from the **`/odometry/filtered`** topic and evaluate its accuracy against the robot's referential position measured by the laser tracker described below.

The **Leica AT403 absolute tracker** will be periodically used for accurately measuring the robot's absolute pose in the global coordinates defined by the tracker itself.

Also the **HTC Vive tracking system** will be used for 2 purposes. First we will try to use it as another referential measurement device providing information about the robot's pose. Second purpose is the automatization of the Leica AT403 absolute tracker's measurement, as colleagues David Štych and Dimitrij Sojma made a set of ROS tools enabling the HTC Vive tracking system to guide the laser tracker to positions where the retro-reflectors are placed on the robot, which will be described below.

The robot is controlled either automatically by the measurement script or manually by the gamepad.

3.2.1 HTC Vive tracker

The **Vive tracker** is a part of HTC's virtual reality (**VR**) **equipment** called **HTC Vive**. It is a device attachable to a physical object in order to track its **6DOF pose**, with the primary aim to project the object's position and orientation to virtual reality. The Vive tracker must be combined with at least **2** so called Vive base stations in order to track its **6DOF pose**. A base station placed on a high tripod will be further referred as "**the lighthouse**". The Vive tracker can be see in the figure 3.2a and the base station (without a tripod) can be seen in the figure 3.2b. Each Vive tracker has its unique USB dongle called the **Watchman dongle**, which enables a PC to communicate wireless with the particular paired Vive tracker and with all the Lighthouses.

Although the original libraries for communication with HTC Vive devices are protected by HTC and available only for use in applications and games at the gaming platform Steam, a group of engineers have **reverse-engineered** the communication and wrote a **set of ROS tools and libraries** enabling to **integrate the tracking system to ROS**[1]. The ROS node is capable of both mutually **calibrating the Vive trackers and lighthouses** and **publishing their 6DOF poses in dedicated topics** in ROS, yet it has some imperfections. It seems that while the original HTC's library maintains seamless tracking of all the Vive devices, this ROS node, despite mostly working without problems, produces errors and refuses to work in certain conditions. This will be discussed further in this chapter.

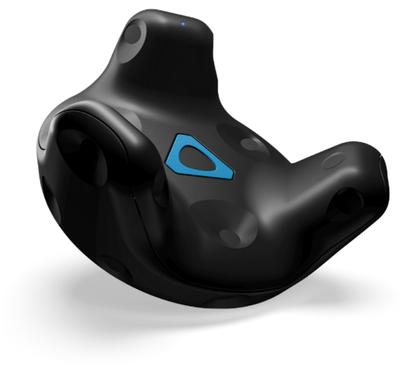
In the further text, the HTC Vive system tracking the pose of the Vive tracker attached to the robot will be referred to as "**the tracking system**".

3.2.2 Leica AT403 absolute tracker

The **Leica AT403 absolute tracker** is a **high-accuracy** portable **laser tracker** able to measure an absolute position of a Leica-trackable object (see below) in its own reference frame. It measures a position of the measured point in spherical coordinates, but it recomputes the measured spherical coordinates of the measured point to the Cartesian coordinates in its own reference frame. In the further text, the Leica absolute tracker will be referred to as "**the laser tracker**".

The laser tracker measures a 3D position of a measurable object distant at least 0.8 m from the laser tracker with accuracy of $\pm(14\,\mu\text{m} + 6\mu\text{m/m})$ ($6\,\mu$m per 1 m of distance from the laser tracker) in each of the 3D coordinates [1, p. 14].

[1]Available at https://github.com/cntools/libsurvive


(a) : HTC Vive tracker (Image taken from https://www.vive.com/us/accessory/vive-tracker/)

(b) : HTC Vive base station 1.0 (Image taken from https://www.vive.com/eu/accessory/base-station/)

Figure 3.2: Used HTC Vive devices

The used Leica-trackable devices are either the retroreflectors mentioned above .

The **retroreflectors** are spherical objects with **retroreflective prism** in its aperture, as is seen in the figures 3.3a and 3.3b. The typical distance in which the retroreflectors are measured without problems is written in [1, p. 14] to be **320 m**, which is beneficial for various applications. There are 2 main types of available retroreflectors. The first type are the retroreflectors for standard application, which can be seen in the figure 3.3a. The second type are the retroreflectors for fixed installation, which can be seen in the figure 3.3b. These retroreflectors do not provide as precise placement of the inner prism as the standard retroreflectors, yet such deep sub-millimeter precision is not needed in the experiments. The use case of the retroreflectors for fixed installation is generally the placement of more than one of them on an object or surface in order to measure its deformation or to study i. e. repeatability of an experiment [2]. Therefore these retroreflectors are used in the experiments, fixed on the robot in the way depicted in the figures 3.4a and 3.4b.

3.2.3 Measuring the robot's pose with the laser tracker guided by the tracking system

During the measurement, we want to guide the laser tracker programmatically so that it does the measurement automatically and the user does not have to guide it manually to the retroreflectors he or she wants to measure. In case of only one retroreflector, there would be no need to guide the laser tracker in any way after having guided it to the reflector (unless it gets covered by an obstacle), because the laser tracker tracks a moving retroreflector automatically by itself. Yet in case of aiming to measure the robot's position and orientation in space by

(a) : Standard Leica retroreflectors [1, p. 14]

(b) : Leica retroreflectors for fixed installation [2]

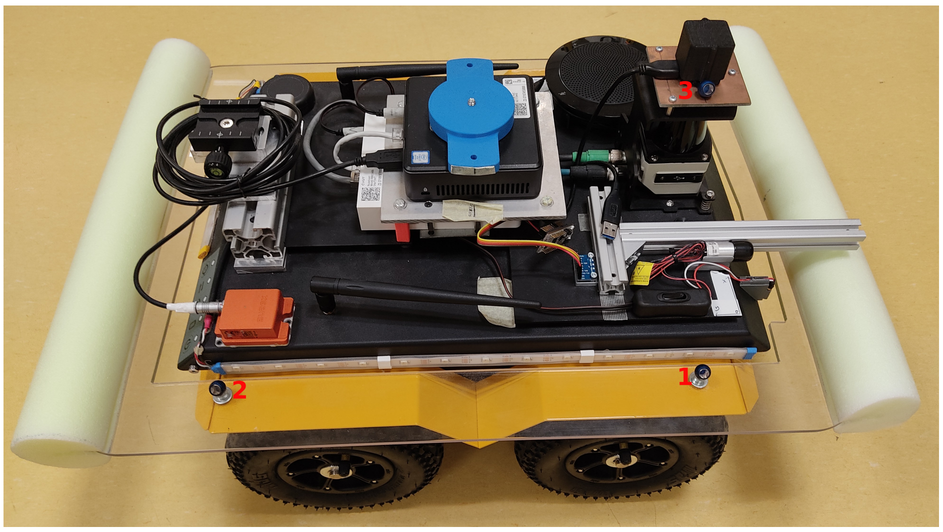
Figure 3.3: Leica-trackable devices

the laser tracker, a group of 3 retroreflectors for fixer installation has been mounted on each the robot's side symetrically (2 groups of 3, thus 6 retroreflectors in total).

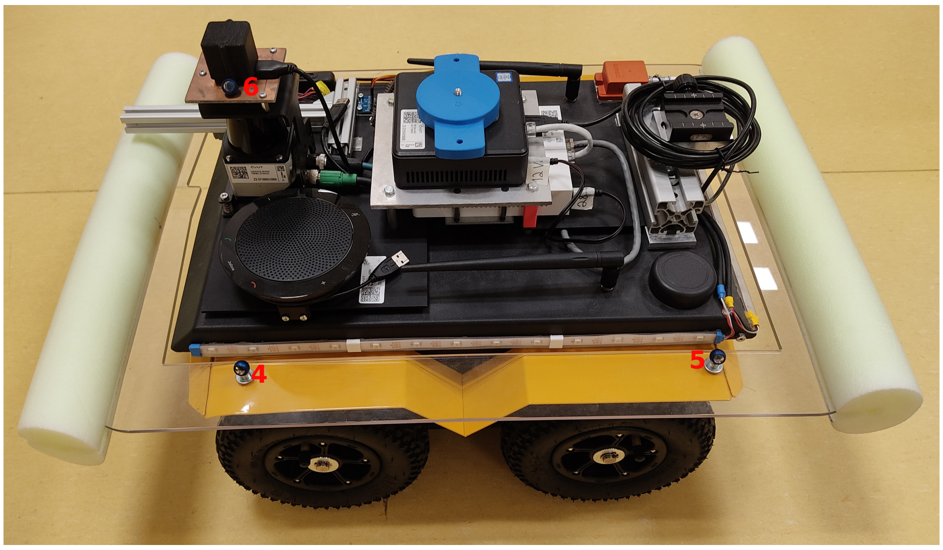
As has been mentioned, for the purpose of measuring the robot's 6DOF pose (position and orientation), colleagues Štych and Sojma wrote a dedicated set of ROS tools[2]. First one of them is a tool for guiding the laser tracker to the wanted lateral (left or right) group of retroreflectors by publishing their estimated 3D positions in the laser tracker's Euclidean coordinate frame in the way described in the Chapter 4, in the section 4.5 on page 36. This tool will be further referred as "**the guidance tool**". The second tool is for the estimation of the transform $\widehat{\boldsymbol{T}}_T^{\mathrm{L}}$, described also in the section 4.5. This tool will be further referred as "**the guidance calibration tool**". The placement of the retroreflectors on the robot and their respective order can be seen in the figures 3.4a and 3.4b.

Štych [40] studied the accuracy of the HTC Vive tracking ROS node and generally the accuracy of the tracking system in combination with the laser tracker. He learned that if the tracking system is not obscurred by external radiation, it tracks the pose very accurately. Yet he learned that the measurement of the laser tracker significantly interferes with the tracking system, especially when trying to find a retroreflector, whose position is sent to it from a computer. Then it interferes so that the poses of the tracking systems are significantly noised and the ROS library for communication with the tracking system may even halt.

[2]Internal package found on CIIRC's GitLab



(a) : Retroreflectors on the right side of the robot an their numbered order



(b) : Retroreflectors on the left side of the robot an their numbered order

Figure 3.4: Placement of the retroreflectors on the robot (with numbered order)

Chapter 4

Theory

4.1 Extended Kalman Filter and the EKF-fused odometry

Reid provides a detailed and comprehensive overview of the probabilistic background, the derivation and basic usage of the EKF in his lecture notes for the subject Estimation [30, 31]. According to his lecture notes, we can generally consider that a system can be represented by a nonlinear discrete-time state-space model in form of

$$\boldsymbol{x}_{k+1} = \mathbf{f}\left(\boldsymbol{x}_k, \boldsymbol{u}_k, k\right) + \boldsymbol{w}_k \,, \tag{4.1}$$

where $\boldsymbol{x}_k$ is the state at time k, $\boldsymbol{u}_k$ is an input control vector, $\boldsymbol{w}_k$ is additive system or process noise, and $\mathbf{f}\left(\boldsymbol{x}_k, \boldsymbol{u}_k, k\right)$ is the **not necessarily linear** state transition vector function [31, p. 39]. We may also consider that the states are observed by certain measurement as

$$\boldsymbol{z}_k = \mathbf{h}\left(\boldsymbol{x}_k, k\right) + \boldsymbol{v}_k \,, \tag{4.2}$$

where $\boldsymbol{z}_k$ is the observation or measurement made at time k , $\boldsymbol{x}_k$ is the state at time k, $\mathbf{h}\left(\boldsymbol{x}_k, k\right)$ is the **nonlinear** observation matrix and $\boldsymbol{v}_k$ is additive measurement noise [31, p. 39]. This observation means the particular **measurement of a sensor**.

Assuming that the **noises** are **Gaussian**, **uncorrelated**, **zero-mean** and **without cross-correlation**, the Extended Kalman Filter has the following optimal form [31, p. 41]. Assuming $\hat{\mathbf{x}}$ as the estimated state vector and $\mathbf{P}$ as the estimated covariance matrix representing the uncertainty of the state vector, the equations are the following.

Prediction:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{f}\left(\boldsymbol{x}_{k|k}, \boldsymbol{u}_k, k\right) , \tag{4.3}$$

$$\mathbf{P}_{k+1|k} = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right] \mathbf{P}_{k|k} \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]^T + \mathbf{Q}_k \,. \tag{4.4}$$

Update:

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}\left[\boldsymbol{z}_{k+1} - \mathbf{h}\left(\hat{\mathbf{x}}_{k+1|k}, k\right)\right] , \tag{4.5}$$

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{K}_{k+1}\mathbf{S}_{k+1}\mathbf{K}_{k+1}^T \,, \tag{4.6}$$

where

$$\boldsymbol{K}_{k+1} = \mathbf{P}_{k+1|k} \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^T \mathbf{S}_{k+1}^{-1} \tag{4.7}$$

is the **Kalman gain matrix**,

$$\mathbf{S}_{k+1} = \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right] \mathbf{P}_{k|k+1} \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right]^T + \mathbf{R}_{k+1} \tag{4.8}$$

is the **innovation covariance matrix**,
$\mathbf{Q}_k$ is the **process noise covariance matrix** at time k and
$\boldsymbol{R}_{k+1}$ is the **measurement noise covariance matrix** at time k+1.

4.2 The robot's kinematic model and its use

The **kinematics** of the robot can be described by the **Skid-Steering kinematic model**, as has been mentioned above.

4.2.1 The Skid-Steering kinematic model

The model used by the controller driving the robot, which is described below in the subsection 5.1.2, does not consider any deeper physical relations such as friction or rolling resistance, in contrast with the model that has been proposed by [29]. The **skid-steering kinematic model** which the controller considers can be expressed as follows.

The mathematical description of this model has been described many times in many publications. Let us thus use the description written in the article of Kozłowski and Pazderski [19].

Let's suppose that we have a skid-steering mobile robot ("**SSMR**") in a global reference frame with the orthonormal basis $\begin{bmatrix}\boldsymbol{X_g} & \boldsymbol{Y_g} & \boldsymbol{Z_g}\end{bmatrix}$. The robot has its center of mass ("**COM**"), which determines its position in the global reference frame by a vector $\boldsymbol{X} = \begin{bmatrix}X & Y & Z\end{bmatrix}^T$. The position of the COM determines the origin of the robot's local reference frame with the orthonormal basis $\begin{bmatrix}\boldsymbol{x_l} & \boldsymbol{y_l} & \boldsymbol{z_l}\end{bmatrix}$, in which coordinates of a point are determined by a vector $\boldsymbol{x} = \begin{bmatrix}x & y & z\end{bmatrix}$. The vector $\boldsymbol{x_l}$ is in the robot's forward direction. The vector $\boldsymbol{y_l}$ is perpendicular to the vector $\boldsymbol{x_l}$ and is in the robot's left lateral direction. The vector $\boldsymbol{z_l}$ is parallel to the vector $\boldsymbol{Z_g}$. We consider only a planar movement of the robot. Therefore we consider the Z coordinate of the robot's COM to be a constant ($Z = const.$). We also denote the angle θ as the planar tilt of the robot, meaning the angle between the vectors $\boldsymbol{X_g}$ and $\boldsymbol{x_l}$. All the reference frames and coordinates are depicted in the figure 4.1.

Combined together, as we consider the robot's planar movement, we may denote a vector of generalized coordinates as $\boldsymbol{q} = \begin{bmatrix}X & Y & \theta\end{bmatrix}^T$ and the corresponding vector of generalized velocities $\dot{\boldsymbol{q}} = \begin{bmatrix}\dot{X} & \dot{Y} & \dot{\theta}\end{bmatrix}^T$. We also define the vector of local linear velocity as $\boldsymbol{v} = \begin{bmatrix}v_x & v_y & 0\end{bmatrix}^T$, where v_x and v_y are the forward and lateral linear velocities, respectively, and the vector of local angular velocity $\boldsymbol{\omega} = \begin{bmatrix}0 & 0 & \omega\end{bmatrix}^T$, where ω is the angular velocity of the robot meaning the angular velocity around the z-axis. All those coordinates and velocities are depicted in the figure 4.2, where c denotes a half of the distance between the centers of the left and of the right pair of wheels (half of the "**wheel separation**") and the meaning

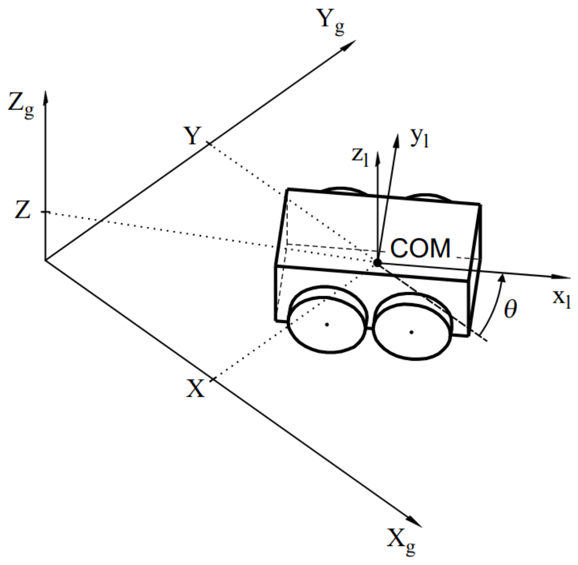


Figure 4.1: Reference frames of a SSMR [19, p. 2]

of the dimensions a, b is depicted. It is then obvious that we may consider the **state-space model** of the robot as

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} . \tag{4.9}$$

Let us now focus on the velocities of the wheels. All the wheels have radius r_i and twist with angular velocity of ω_i, where $i \in \{1, \dots, N\}$ and N is the count of the wheels. Therefore the forward velocity of each of the N wheels can be computed as

$$v_{ix} = \omega_i r_i . \tag{4.10}$$

Different velocities of the wheels may create either straightforward movement or circular movement of the robot, determined by its instantaneous center of rotation ("**ICR**") with coordinates in the robot's local reference frame $\begin{bmatrix} x_{ICR} & y_{ICR} & 0 \end{bmatrix}^T$. It is depicted in the figure 4.3, where $\boldsymbol{v_i} = \begin{bmatrix} v_{ix} & v_{iy} & 0 \end{bmatrix}^T$ is the vector of local linear velocities of the i-th wheel, $\boldsymbol{P_i}$ is the position of its center and d_i is the distance of its center from the ICR. The distance of the COM from the ICR is there denoted as d_C.

In this work we assume a 4-wheeled SSMR, whose model is depicted in the figures 4.1 through 4.3. We also neglect the lateral velocities and we assume that both the left wheels have the same angular velocity $\omega_L = \omega_1 = \omega_2$, because they are bounded, as well as both the right wheels have the same angular velocity $\omega_R = \omega_3 = \omega_4$. Assuming that all the wheels have the same radius r, according to Kozłowski and Pazderski [19], we may express the control

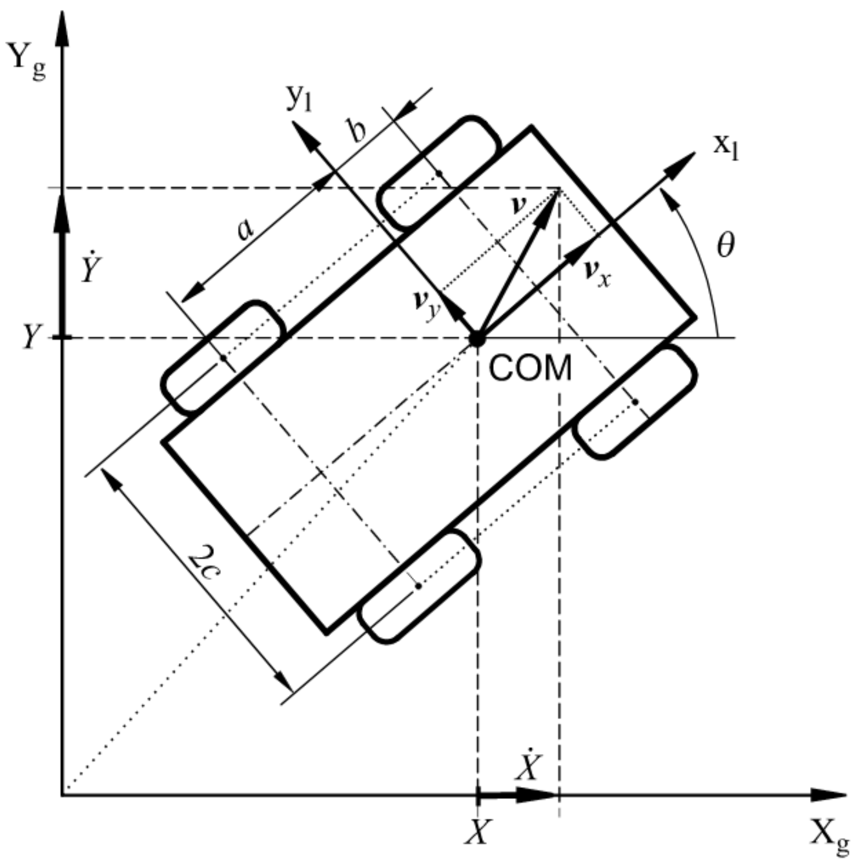


Figure 4.2: Planar reference frames and generalized coordinates and velocities of a SSMR[19, p. 2]

input vector

$$\boldsymbol{\eta} = \begin{bmatrix} v_x \\ \omega \end{bmatrix} = r \begin{bmatrix} \frac{\omega_L + \omega_R}{2} \\ \frac{-\omega_L + \omega_R}{2c} \end{bmatrix} , \tag{4.11}$$

where v_x is the commanded linear velocity, ω is the commanded angular velocity of the robot and $2c$ is the wheel separation mentioned above. We may thus express the dependence of the angular velocities of the lateral pairs of wheels on the control input from the equation (4.11) as

$$\begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix} = \begin{bmatrix} \frac{1}{r}\left(v_x - c\omega\right) \\ \frac{1}{r}\left(v_x + c\omega\right) \end{bmatrix} . \tag{4.12}$$

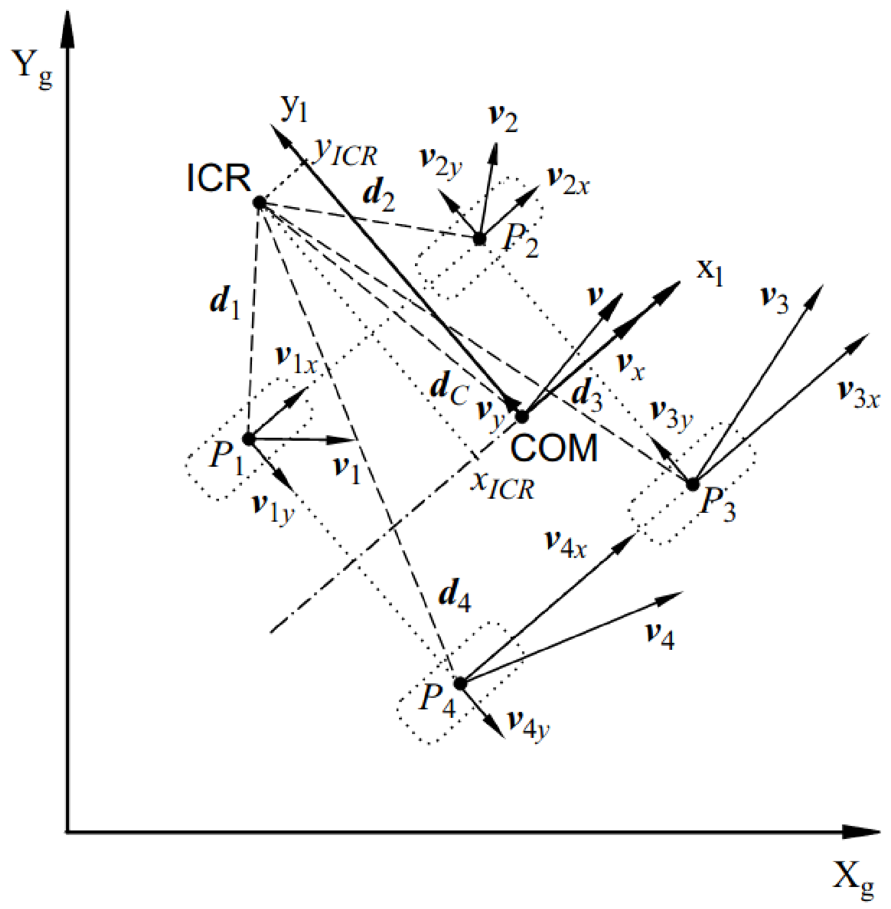


Figure 4.3: Planar reference frames and generalized coordinates and velocities of a SSMR[19, p. 2]

4.2.2 The controller's mathematical background

the robot uses the **diff_drive_controller** ROS package, which is described in the Implementation chapter in the subsection 5.1.2 on page 45. It is obvious from the controller's code written by Magyar [25, downloaded package] that besides the velocity, acceleration and jerk limiting, it performs the following computations.

Having applied the optional limits, it first computes the **adjusted wheel separation** as

$$s^* = k_s \cdot s_0 \,, \tag{4.13}$$

where s_0 is the real wheel separation ($2c$) taken from the **wheel_separation** parameter and k_s is the wheel separation multiplier taken from the **wheel_separation_multiplier** parameter. Then it computes the **adjusted wheel radius** as

$$r^* = k_r \cdot r_0 \,, \tag{4.14}$$

where r_0 is the real wheel radius taken from the **wheel_radius** parameter and k_r is the wheel separation multiplier taken from the **wheel_radius_multiplier** parameter.

After the adjusted wheel separation and the adjusted wheel radius used for the approximation as ideal differentially driven robot are computed, the controller computes the resulting **left and right commanded wheel angular velocities** as

$$\begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix} = \begin{bmatrix} \frac{1}{r^*}\left(v_x - \frac{s^*}{2}\omega\right) \\ \frac{1}{r^*}\left(v_x + \frac{s^*}{2}\omega\right) \end{bmatrix} \,, \tag{4.15}$$

where v_x is the wanted limited forward velocity and ω_z is the wanted limited angular velocity of the robot.

It is apparent that the controller uses the model explained above in the subsection 4.2.1. It only applies the **wheel radius coefficient** k_r to compute the **adjusted wheel radius**, which may physically express for example **non-standard inflation of the wheels**, and the **wheel separation coefficient** k_s to compute the **adjusted wheel separation**, a half of which represents the absolute value of the local y-coordinate of the ICR's. Indeed, if we compare the equations (4.12) and (4.15), we obtain the substitution relation

$$c = \frac{k_s \cdot s}{2} \,, \tag{4.16}$$

which is applied by the controller to the equation (4.12).

4.3 Boxan's extension of Nováček's NDT SLAM

Nováček [27] and Boxan [4] use the **Normal distribution transform** (**NDT**) for matching the point cloud scanned by the LiDAR to the map. The idea behind it is following.

First the map is split into a grid of identically formed square cells. Then the points in the map contained in each of the cells are approximated with a Gaussian in each cell. Each cell of the map is thus represented by the mean of the points $\boldsymbol{\mu}$ and their respective covariance matrix $\boldsymbol{\Sigma}$. After each scan of the LiDAR, the resulting point cloud is fitted in the NDT grid by optimizing the score function evaluating the accuracy of the match between the point cloud and the map for the pose, which is being found by the optimization of the score function.

Nováček's score function [27, p. 14] is expressed by Boxan [4, p. 5] as

$$s(\boldsymbol{p}) = \sum_i \exp\left(-\frac{1}{2}\left(\boldsymbol{x}_i' - \boldsymbol{\mu}_i\right)^T \boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{x}_i' - \boldsymbol{\mu}_i\right)\right), \tag{4.17}$$

where $\boldsymbol{p} = \begin{bmatrix} t_x & t_y & \theta \end{bmatrix}^T$ is the vector coordinates representing the matched scan's alignment in the map, $\boldsymbol{x}_i'$ is the i-th scanned point transformed by the transformation

$$\boldsymbol{x}_i' = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \boldsymbol{x}_i + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{4.18}$$

and $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the mean and covariance matrix, respectively, of the map's NDT grid cell corresponding to the transformed scanned point $\boldsymbol{x}_i'$.

The score function $s(\boldsymbol{p})$ is then optimized by minimizing its negative value $-s(\boldsymbol{p})$ iteratively by the Newton's algorithm, solving the equation

$$\boldsymbol{\mathcal{H}}\Delta\boldsymbol{p} = -\boldsymbol{g}\,, \tag{4.19}$$

where $\boldsymbol{\mathcal{H}}$ and $\boldsymbol{g}$ are the Hessian matrix and gradient, respectively, detailed information about which can be found in Nováček's work [27, p. 14-15], as well as the information about the NDT.

Boxan approximates Nováček score function with a multivariate normal distribution around the pose taken from the EKF-fused odometry originating in the pose obtained from the algorithm's previous result as [4, p. 13]:

$$M(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = s(\boldsymbol{p})|_{\mu} \exp\left(-\frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x} - \boldsymbol{\mu}\right)\right), \tag{4.20}$$

where $\boldsymbol{x}$ is the variable representing the robot's pose $\boldsymbol{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$ representing the scan match and $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the estimated pose from the odometry and its covariance matrix, respectively. For the improved optimisation he uses the **improved score function** consisting of a sum of (4.17) and (4.20):

$$s^*(\boldsymbol{x}) = s(\boldsymbol{x}) + M(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma})\,, \tag{4.21}$$

where $\boldsymbol{x}$ is the searched robot's pose. This function tends to look more like a single peak being found, even in a long narrow corridor. For its better performance and smaller computational complexity Boxan also considers the visibility of objects from the current estimated pose.

After each run of the algorithm it is checked whether the determinant of the Hessian matrix of the improved score function is negative or not. In case that it is negative, the new covariance matrix is set as

$$\boldsymbol{\Sigma_t} = -\boldsymbol{\mathcal{H}}\left(s(\boldsymbol{p_i})|_t + M(\boldsymbol{p_i}, \boldsymbol{\mu}, \boldsymbol{\Sigma})|_t\right)\,, \tag{4.22}$$

where $\boldsymbol{p_i}$ is the final pose found by the algorithm. Otherwise, the covariance matrix is left-multiplied by a matrix $\boldsymbol{C}$, values of which have been set temporarily to the guessed ones.

Now it remains to accurately model the odometry uncertainty along with its evolution over time and the trajectory and to improve the uncertainty consideration in the SLAM algorithm.

4.4 Finding a transform between 2 sets of corresponding points

Sometimes we need to match 2 corresponding sets of points by finding a transform between them. The method for finding it mentioned by Ho [13] can be expressed using homogeneous coordinates as follows.

Assume that we have 2 matrices of n 3D points, where necessarily $n \geq 3$, expressed in homogeneous coordinates as

$$\boldsymbol{A} = \begin{bmatrix} x_{a1} & \cdots & x_{an} \\ y_{a1} & \cdots & y_{an} \\ z_{a1} & \cdots & z_{an} \\ 1 & \cdots & 1 \end{bmatrix}, \tag{4.23}$$

$$\boldsymbol{B} = \begin{bmatrix} x_{b1} & \cdots & x_{bn} \\ y_{b1} & \cdots & y_{bn} \\ z_{b1} & \cdots & z_{bn} \\ 1 & \cdots & 1 \end{bmatrix}, \tag{4.24}$$

where $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{4\times n}$ are the matrices of homogeneous coordinates of the respective sets of points. We want to find a homogemeous transform

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{4.25}$$

between them, with rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$, such that

$$\boldsymbol{T}\boldsymbol{A} = \boldsymbol{B}\,. \tag{4.26}$$

If the points are noisy, we want to find $\boldsymbol{T}$, which minimizes the least squares error

$$e = \sum_{i=1}^{n} \left\| \boldsymbol{T}\boldsymbol{A}^i - \boldsymbol{B}^i \right\|^2, \tag{4.27}$$

where $\boldsymbol{A}^i$ and $\boldsymbol{B}^i$ are the i-th columns of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively, and n is the count of the points in each of the corresponding sets.

In order to find it, we first define the non-homogeneous representation of the points in the matrices as

$$\boldsymbol{A}_3^i = \begin{bmatrix} x_{ai} & y_{ai} & z_{ai} \end{bmatrix}^T, \tag{4.28}$$

$$\boldsymbol{B}_3^i = \begin{bmatrix} x_{bi} & y_{bi} & z_{bi} \end{bmatrix}^T, \tag{4.29}$$

which together create the matrices of non-homogeneous points $\boldsymbol{A}_3$ and $\boldsymbol{B}_3$, respectively. Then we compute the centroids

$$\boldsymbol{c_A} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{A}_3^i\,, \tag{4.30}$$

$$\boldsymbol{c_B} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{B}_3^i\,. \tag{4.31}$$

Next we compute the covariance matrix $\boldsymbol{H}$ as

$$\boldsymbol{H} = (\boldsymbol{A} \ominus \boldsymbol{c_A})\,(\boldsymbol{B} \ominus \boldsymbol{c_B})\ , \tag{4.32}$$

where the operator $\boldsymbol{X} \ominus \boldsymbol{y}$ denotes a subtraction of a vector $\boldsymbol{y}$ from each of the columns of a matrix $\boldsymbol{X}$. Subsequently we apply the SVD decomposition to the matrix $\boldsymbol{H}$, obtaining matrices $\boldsymbol{U}$, $\boldsymbol{S}$ and $\boldsymbol{V}$:

$$[\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}] = SVD\,(\boldsymbol{H})\ . \tag{4.33}$$

Now, we obtain the rotation matrix $\boldsymbol{R}$ as

$$\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}^T\,. \tag{4.34}$$

It is then necessary to check, whether a reflection matrix was not computed, which would be a non-sense in the real world. So, if $\det(\boldsymbol{R}) < 0$, it means that the $\boldsymbol{R}$ is a reflection matrix. Therefore the computation of $\boldsymbol{R}$ must be adjusted in the following way. The 3rd column of $\boldsymbol{V}$ has to be multiplied by -1 and then the $\boldsymbol{R}$ is computed like in (4.34). Put all together, the adjusted equation (4.34) can be expressed as

$$\boldsymbol{R} = \begin{cases} \boldsymbol{V}\boldsymbol{U}^T & \det\left(\boldsymbol{V}\boldsymbol{U}^T\right) \geq 0 \\ \boldsymbol{V}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}\boldsymbol{U}^T & otherwise \end{cases}\,. \tag{4.35}$$

Finally we can compute the translation vector $\boldsymbol{t}$ as

$$\boldsymbol{t} = \boldsymbol{c_B} - \boldsymbol{R}\boldsymbol{c_A}\,. \tag{4.36}$$

4.5 Estimating poses of the retroreflectors for guiding the laser tracker using the tracking system

The laser tracker and the tracking system are described in the Chapter 6, in the subsections 3.2.1 on page 20 and 3.2.2 on page 20, respectively. Their relation in the experiments is then described in the subsection 3.2.3 on page 21. The tracking system is used to guide the laser tracker to positions of a triplet out of the 6 retroreflectors (described also in the subsection 3.2.2 on page 20) mounted on the robot, which are estimated by the tracking system. The placement of the retroreflectors on the robot and their respective order can be seen in the figures 3.4b and 3.4a on page 23. Let us now study the way the positions of the retroreflectors in the reference frame of the laser tracker are estimated from the pose estimated by the tracking system by the tool written by colleagues Štych and Sojma. This tool will be further referred as "**the guidance tool**". They also wrote a tool for estimation of a crucial transform needed in this tool, which will be described below. This tool will be further referred as "**the guidance calibration tool**".

The reference frames and transforms are depicted in the diagram in the figure 4.4. The referential coordinate frames are all orthonormal Cartesian coordinate frames and are denoted as: **W** (the world coordinate frame), **L** (Cartesian coordinate frame of the laser tracker), **V** (the coordinate frame of the tracking system (V for Vive)) and **T** (the local coordinate frame of the HTC Vive Tracker attached to the robot). The poses of the retroreflectors are expressed by the following vectors. For the i-th retroreflector ($i \in \{1, \dots, 6\}$), the vector $\boldsymbol{r}_{i\mathrm{L}}$ denotes its position in the coordinate frame **L** (of the laser tracker), which is measured by the laser tracker, and the vector $\boldsymbol{r}_{i\mathrm{T}}$ denotes its position in the coordinate frame **T** (of the tracker attached to the robot), which is constant and known from a manual measurement. Generally a transform $\boldsymbol{T}^{\mathrm{B}}_{\mathrm{A}}$ denotes a homogeneous transform from a coordinate frame A to a coordinate frame B. The transforms $\boldsymbol{T}^{\mathrm{L}}_{\mathrm{W}}$ and $\boldsymbol{T}^{\mathrm{V}}_{\mathrm{W}}$ are not known, but they are not needed and thus are ignored. The transform $\boldsymbol{T}^{\mathrm{V}}_{\mathrm{T}}$ is obtained from the tracking system, yet primarily as a 6DOF pose expressed by a set of a translation vector and a quaternion expressing the rotation. Finally, the transforms $\boldsymbol{T}^{\mathrm{L}}_{\mathrm{V}}$ and thus also $\boldsymbol{T}^{\mathrm{L}}_{T}$ are primarily unknown, yet we can express the latter as

$$\boldsymbol{T}^{\mathrm{L}}_{T} = \boldsymbol{T}^{\mathrm{L}}_{\mathrm{V}} \boldsymbol{T}^{\mathrm{V}}_{\mathrm{T}} \,. \tag{4.37}$$

Therefore the only important unknown transform is the $\boldsymbol{T}^{\mathrm{L}}_{\mathrm{V}}$, which is needed to be estimated.

The problem is to estimate the vectors $\boldsymbol{r}_{1\mathrm{L}}$ through $\boldsymbol{r}_{6\mathrm{L}}$ from the pose of the frame **T** in the frame **V** denoted by the transform $\boldsymbol{T}^{\mathrm{V}}_{\mathrm{T}}$, obtained from the tracking system, so that the laser tracker can be guided to measure their accurate position within its frame **L**. The guidance tool solves this problem using the mathematical tool described in the section 4.4 for subsequent estimation of the transform $\boldsymbol{T}^{\mathrm{L}}_{\mathrm{V}}$. It uses the known vectors $\boldsymbol{r}_{1\mathrm{T}}$ through $\boldsymbol{r}_{6\mathrm{T}}$ and measured positions of a triplet out of 6 retroreflectors mounted on the robot to find a particular transforms as the transform best aligning 2 sets of 3 points.

First, keeping the robot still, positions of a triplet of retroreflectors, either of the first 3 or the last 3, are gradually manually measured by the laser tracker with help of a user, depending on which side of the robot faces the laser tracker. At the same time, the transform $\boldsymbol{T}_{\mathrm{T}}^{\mathrm{V}}$ obtained from the 6DOF pose obtained from the tracking system is saved as $\boldsymbol{T'}_{\mathrm{T}}^{\mathrm{V}}$. Then, in order to estimate the transform $\boldsymbol{T}_{\mathrm{V}}^{\mathrm{L}}$, the equations (4.23) through (4.36) are used to estimate the transform $\boldsymbol{T}_{\mathrm{V}}^{\mathrm{L}}$ from the triplet of known constant vectors $\left(\boldsymbol{r}_{i_0\mathrm{T}}, \boldsymbol{r}_{(i_0+1)\mathrm{T}}, \boldsymbol{r}_{(i_0+2)\mathrm{T}}\right)$ and the triplet of measured vectors $\left(\boldsymbol{r}_{i_0\mathrm{L}}, \boldsymbol{r}_{(i_0+1)\mathrm{L}}, \boldsymbol{r}_{(i_0+2)\mathrm{L}}\right)$, where $i_0 = 1$ or $i_0 = 4$. There, if we consider the notation of vectors

$$\boldsymbol{r}_{i\mathrm{T}} = \begin{bmatrix} x_{i\mathrm{T}} & y_{i\mathrm{T}} & z_{i\mathrm{T}} \end{bmatrix}^T , \tag{4.38}$$

$$\boldsymbol{r}_{i\mathrm{L}} = \begin{bmatrix} x_{i\mathrm{L}} & y_{i\mathrm{L}} & z_{i\mathrm{L}} \end{bmatrix}^T , \tag{4.39}$$

the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, expressed by the equations (4.23) and (4.24), respectively, can be substituted by the matrices

$$\boldsymbol{A} = \begin{bmatrix} x_{i_0\mathrm{T}} & x_{(i_0+1)\mathrm{T}} & x_{(i_0+2)\mathrm{T}} \\ y_{i_0\mathrm{T}} & y_{(i_0+1)\mathrm{T}} & y_{(i_0+2)\mathrm{T}} \\ z_{i_0\mathrm{T}} & z_{(i_0+1)\mathrm{T}} & z_{(i_0+2)\mathrm{T}} \\ 1 & 1 & 1 \end{bmatrix} , \tag{4.40}$$

$$\boldsymbol{B} = \begin{bmatrix} x_{i_0\mathrm{L}} & x_{(i_0+1)\mathrm{L}} & x_{(i_0+2)\mathrm{L}} \\ y_{i_0\mathrm{L}} & y_{(i_0+1)\mathrm{L}} & y_{(i_0+2)\mathrm{L}} \\ z_{i_0\mathrm{L}} & z_{(i_0+1)\mathrm{L}} & z_{(i_0+2)\mathrm{L}} \\ 1 & 1 & 1 \end{bmatrix} . \tag{4.41}$$

Using these matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ as substitution in the equations (4.23) through (4.36), the transform $\boldsymbol{T}_{\mathrm{T}}^{\mathrm{L}}$ is estimated as $\widehat{\boldsymbol{T}}_{\mathrm{T}}^{\mathrm{L}}$ by the rotation matrix and translation vector, expressed in the equations (4.35) and (4.36), respectively. From there, expressed from the equation (4.37), the estimation of the wanted transform $\boldsymbol{T}_{\mathrm{V}}^{\mathrm{L}}$ is computed as

$$\widehat{\boldsymbol{T}}_{\mathrm{V}}^{\mathrm{L}} = \widehat{\boldsymbol{T}}_{\mathrm{T}}^{\mathrm{L}} \left(\boldsymbol{T'}_{\mathrm{T}}^{\mathrm{V}}\right)^{-1} , \tag{4.42}$$

where the transform $\boldsymbol{T'}_{\mathrm{T}}^{\mathrm{V}}$ is the transform $\boldsymbol{T}_{\mathrm{T}}^{\mathrm{V}}$ obtained from the 6DOF pose, which was published by the tracking system at the time right before the measurement of the position of the first retroreflector in the measured triplet. All this calibration is maintained by **the guidance calibration tool**.

This estimated transform $\widehat{\boldsymbol{T}}_{\mathrm{V}}^{\mathrm{L}}$ is then used in order to estimate the positions of all the 6 retroreflectors in the reference frame **L** in real time. Subsequently, the estimated transform

$$\widehat{\boldsymbol{T}}_{T}^{\mathrm{L}} = \widehat{\boldsymbol{T}}_{\mathrm{V}}^{\mathrm{L}} \boldsymbol{T}_{\mathrm{T}}^{\mathrm{V}} , \tag{4.43}$$

where $\boldsymbol{T}_{\mathrm{T}}^{\mathrm{V}}$ is the actual transform obtained from the actual 6DOF pose obtained from the tracking system, is applied on the constant known vectors $\boldsymbol{r}_{1\mathrm{T}}$ through $\boldsymbol{r}_{6\mathrm{T}}$, transformed to homogeneous 3-dimensional vectors as $\widetilde{\boldsymbol{r}}_{1\mathrm{T}}$ through $\widetilde{\boldsymbol{r}}_{6\mathrm{T}}$. Then, the estimated positions of the retroreflectors in the coordinate frame of the laser tracker, **L**, expressed as homogeneous vectors are obtained as

$$\widehat{\widetilde{\boldsymbol{r}}}_{i\mathrm{L}} = \widehat{\boldsymbol{T}}_T^{\mathrm{L}} \widetilde{\boldsymbol{r}}_{i\mathrm{T}}\,, \qquad i \in \{1, \ldots, 6\}\ , \tag{4.44}$$

from where we obtain the estimated coordinates of each retroreflector $\widehat{\boldsymbol{r}}_{1\mathrm{L}}$ through $\widehat{\boldsymbol{r}}_{6\mathrm{L}}$. These estimated positions are continuously published by **the guidance tool** to guide the laser tracker.

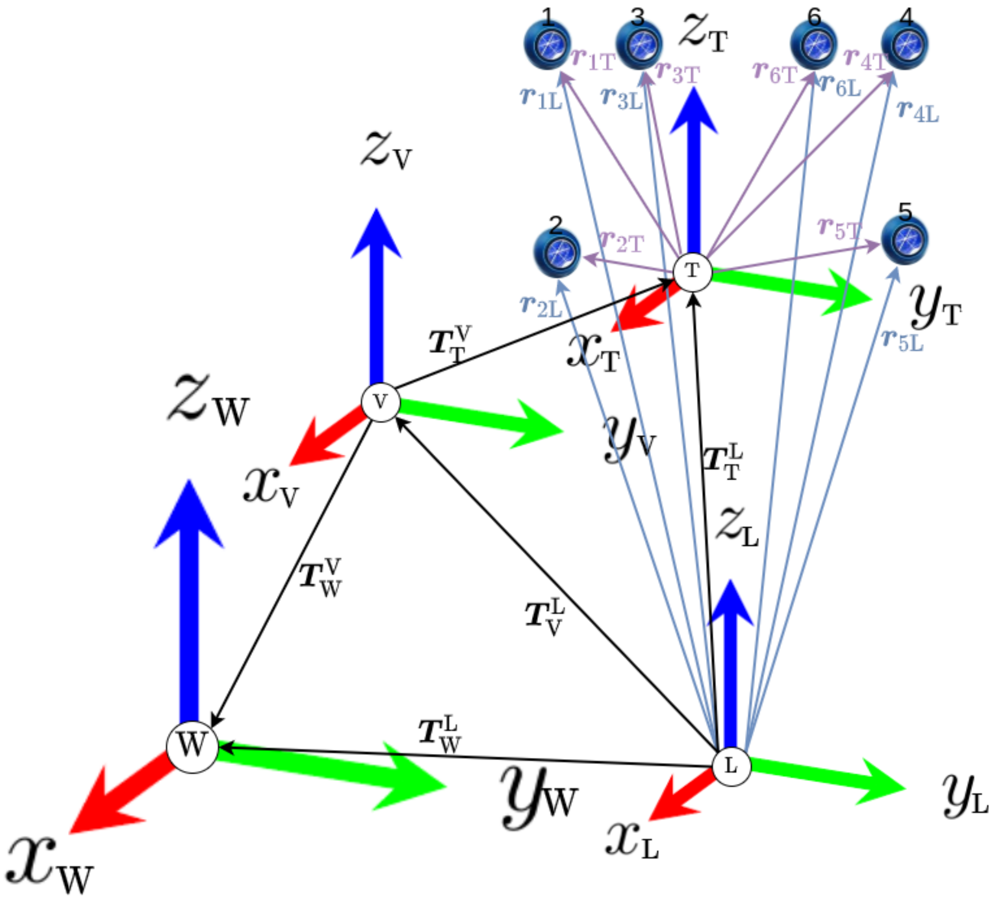


Figure 4.4: Diagram of referential coordinate frames, transforms between them and positions of retroreflectors towards the referential frames. The referential coordinate frames are all orthonormal Cartesian coordinate frames and are denoted as: **W** (the world coordinate frame), **L** (Cartesian coordinate frame of the laser tracker), **V** (the coordinate frame of the tracking system (V for Vive)) and **T** (the local coordinate frame of the HTC Vive Tracker attached to the robot). The poses of the retroreflectors are expressed by the following vectors. For the i-th retroreflector ($i \in \{1, \ldots, 6\}$), the vector $\boldsymbol{r}_{i\mathrm{L}}$ denotes its position in the coordinate frame **L** (of the laser tracker), which is measured by the laser tracker, and the vector $\boldsymbol{r}_{i\mathrm{T}}$ denotes its position in the coordinate frame **T** (of the tracker attached to the robot), which is constant and known from a manual measurement.

4.6 Modeling the uncertainty of the robot's EKF-filtered odometry

In the experiment described in the Chapter 6 in the section **??** on page ??, we need first to model the uncertainty of the EKF-filtered odometry by a mean error vector and an error covariance matrix, based on the experimental data and the commanded velocities and driven distances. Subsequently, we need to approximate the contribution to the mean error vector and the error covariance matrix based on the commanded forward and angular velocities and the increment of the driven distance. All that is needed in order to make the SLAM more accurate. Here are the steps needed to render it.

4.6.1 Modeling of the mean error vector and covariance matrix from the experimental data

Let us assume that for each final pose of each evaluated odometry, which is further described in the section 6.4 on page 77, we have:

- Assumed **the coordinate system in the originating 3DOF null pose** of the odometry represented by a two-dimensional homogeneous transformation matrix

$$\boldsymbol{T_{OO}} = \mathbb{E}_{\mathbf{3}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.45}$$

- Saved **the coordinate system in the final 3DOF null pose** $\boldsymbol{x_F} = \begin{bmatrix} x_F & y_F & \theta_F \end{bmatrix}^T$ subsequently represented by a two-dimensional homogeneous transformation matrix

$$\boldsymbol{T_{OF}} = \begin{bmatrix} \cos(\theta_F) & -\sin(\theta_F) & x_F \\ \sin(\theta_F) & \cos(\theta_F) & y_F \\ 0 & 0 & 1 \end{bmatrix} \tag{4.46}$$

- Measured **triplet of retroreflectors' positions in the beginning** of actually studied trajectory $(\boldsymbol{x_{b1}}, \boldsymbol{x_{b2}}, \boldsymbol{x_{b3}}) : \boldsymbol{x_{bi}} = \begin{bmatrix} x_{bi} & y_{bi} & z_{bi} \end{bmatrix}^T$
- Measured **triplet of retroreflectors' positions in the end** of actually studied trajectory $(\boldsymbol{x_{e1}}, \boldsymbol{x_{e2}}, \boldsymbol{x_{e3}}) : \boldsymbol{x_{ei}} = \begin{bmatrix} x_{ei} & y_{ei} & z_{ei} \end{bmatrix}^T$

Since we want to study the uncertainty of the robot's odometry only in terms of the planar 3DOF pose, we may transform the triplets of measured retroreflectors' positions into transforms expressing the robot's referential poses in the beginning and at the end of the studied partial partial trajectory the following way. We may assume

- the 3rd retroreflector's xy-coordinates as the robot's xy-coordinates *and*
- the xy-coordinates of the vector leading from the 2nd to the 1st retroreflector as the vector of the robot's orientation

Then the 2D homogeneous transforms describing coordinate systems representing the robot's referential poses from Leica measurements can be expressed as

$$\boldsymbol{T_{LO}} = \begin{bmatrix} \cos(\theta_{LO}) & -\sin(\theta_{LO}) & x_{b3} \\ \sin(\theta_{LO}) & \cos(\theta_{LO}) & y_{b3} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.47}$$

$$\boldsymbol{T_{LF}} = \begin{bmatrix} \cos(\theta_{LF}) & -\sin(\theta_{LF}) & x_{e3} \\ \sin(\theta_{LF}) & \cos(\theta_{LF}) & y_{e3} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.48}$$

where

$$\theta_{LO} = \operatorname{arctan2}\left(y_{b1} - y_{b2}, x_{b1} - x_{b2}\right) \tag{4.49}$$

$$\theta_{LF} = \operatorname{arctan2}\left(y_{e1} - y_{e2}, x_{e1} - x_{e2}\right) \tag{4.50}$$

Having expressed all the transforms, we may now express the robot's absolute 3DOF pose error as follows. We express the **homogeneous transform representing the robot's referential pose**, taken from Leica's measurements, as if Leica began in the null pose in order to reasonably compare Leica and odometry final poses (because the odometry also begins in the null pose) as

$$\boldsymbol{T_L} = \boldsymbol{T_{LO}^{-1}} \boldsymbol{T_{LF}} \tag{4.51}$$

This transform represents the robot's referential pose in the end of the studied trajectory.

Now we want to express the error of the odometry by comparing the transform describing its final pose $\boldsymbol{T_O} = \boldsymbol{T_{OF}}$ (because $\boldsymbol{T_{OO}}$ is an unit matrix) to Leica's $\boldsymbol{T_L}$. Let us consider a function which transforms a 2D homogeneous transform matrix to a 3DOF planar pose

$$\boldsymbol{p}\left(\begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ 0 & 0 & 1 \end{bmatrix}\right) = \begin{bmatrix} T_{13} \\ T_{23} \\ \operatorname{arctan2}\left(T_{21}, T_{11}\right) \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_\theta \end{bmatrix} \tag{4.52}$$

Now we may express the **absolute error of the odometry** as

$$\boldsymbol{e_{abs}} = \boldsymbol{p}\left(\boldsymbol{T_O}\right) - \boldsymbol{p}\left(\boldsymbol{T_L}\right) \tag{4.53}$$

yet mainly we want to **study the error seen relatively from the view in the robot's referential pose**. Therefore we express the **relative error of the odometry** as

$$\boldsymbol{e} = \boldsymbol{p}\left(\boldsymbol{T}_{\boldsymbol{L}}^{-1}\boldsymbol{T}_{\boldsymbol{O}}\right) \tag{4.54}$$

As (at least partial) result of this measurement we want to estimate the **mean vector** $\boldsymbol{\mu}_{\boldsymbol{e}}$ and the **covariance matrix** $\boldsymbol{\Sigma}_{\boldsymbol{e}}$ of the error of the robot's **raw** and **EKF-filtered** (also referred to as EKF-fused) **odometries** and **how they evolve based on the robot's forward and angular speed** (and thus also the direction of its drive). They can be expressed for each of the odometries as

$$\boldsymbol{\mu}_{\boldsymbol{e}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{e} \tag{4.55}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{e}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{e}^{\boldsymbol{T}}\boldsymbol{e} \tag{4.56}$$

where n denotes the count of measured samples for the current trajectory.

4.6.2 Approximation of the contribution to the mean error vector and to the covariance matrix depending on the velocities and the increment of driven distance

The mean error vectors and error covariance matrices corresponding to the data from trajectories driven with properly initialized IMU have been computed for each of the trajectories except the trajectory of forward velocities $v_x = 0.5$ m/s and $v_\theta = -0.1$ rad/s. They were generally expressed as $\boldsymbol{\mu}_{\boldsymbol{e},\boldsymbol{i}}(v_{x,i}, v_{\theta,i}, \text{true})$ and $\boldsymbol{\Sigma}_{\boldsymbol{e},\boldsymbol{i}}(v_{x,i}, v_{\theta,i}, \text{true})$ by the expressions (6.3) through (6.30), where

$$i \in \{1, \ldots, N\} \tag{4.57}$$

will be referred to as the index, where $\boldsymbol{N}$ is the count of the suitable 3 meters long trajectories driven at certain commanded forward velocity v_x and angular velocity v_θ. Suitable trajectories are considered those driven with the IMU properly initialized. The dependency of the values of the components of the mean error vector and error covariance matrix on the given velocities can be therefore now approximated by quadratic functions for each of the mentioned components.

All of the trajectories were driven at the forward velocity of 0.5 m/s (-0.5 m/s when driving backward). Therefore the dependency of the values on the linear velocity will not be expressed numerically, yet the values will be distinguished by the sign of the forward velocity meaning the forward (when positive) or backward (when negative) drive. Let us then compute the values of the coefficients of the resulting quadratic terms for each of the components, expressing the dependency of the values of the components on the commanded angular velocity.

The values of the coefficients can be simply approximated by means of the **least squares** method, generally approximating a solution of the matrix equation

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{B} \tag{4.58}$$

for a variable matrix $\boldsymbol{X}$, where $\boldsymbol{A}$ and $\boldsymbol{B}$ are known matrices, in terms of the least sum of the 2-norms of the error vectors expressing the distance of real values from the approximated ones.

In this case, we may make use of the knowledge that the (error) covariance matrices are symmetrical. They can be thus generally expressed as

$$\boldsymbol{\Sigma_{e,i}} = \begin{bmatrix} m_{1,i} & m_{2,i} & m_{3,i} \\ m_{2,i} & m_{4,i} & m_{5,i} \\ m_{3,i} & m_{5,i} & m_{6,i} \end{bmatrix} , \tag{4.59}$$

which means that a covariance matrix can be equally represented by a vectorized form of its upper diagonal block. If we then express an error vector as

$$\boldsymbol{\mu_{e,i}} = \begin{bmatrix} e_{x,i} & e_{y,i} & e_{\theta,i} \end{bmatrix}^T \tag{4.60}$$

where $e_{x,i}$, $e_{y,i}$ and $e_{\theta,i}$, respectively, are the means of errors of the x-, y- and θ-components, respectively, we may define a row function creating a row of the $\boldsymbol{B}$ matrix, corresponding to a pair of an error mean vector $\boldsymbol{\mu_{e,i}}$ and error covariance matrix $\boldsymbol{\Sigma_{e,i}}$, for the purposes of the least squares as

$$\boldsymbol{r_B}\left(\boldsymbol{\mu_{e,i}}, \boldsymbol{\Sigma_{e,i}}\right) = \begin{bmatrix} e_{x,i} & e_{y,i} & e_{\theta,i} & m_{1,i} & m_{2,i} & m_{3,i} & m_{4,i} & m_{5,i} & m_{6,i} \end{bmatrix} \tag{4.61}$$

Subsequently, we will want to approximate generally coefficients a_Y, b_Y and c_Y for quadratic terms of the general form

$$Y = a_Y v_\theta^2 + b_Y v_\theta + c_Y \tag{4.62}$$

where

$$Y \in \left\{ e_x, \quad e_y, \quad e_\theta, \quad m_1, \quad m_2, \quad m_3, \quad m_4, \quad m_5, \quad m_6 \right\} \tag{4.63}$$

for every variable Y, wherefore we may define a row function creating a row of the $\boldsymbol{A}$ matrix, corresponding to a value of the angular velocity v_θ, as

$$\boldsymbol{r_A}(v_{\theta,i}) = \begin{bmatrix} v_{\theta,i}^2 & v_{\theta,i} & 1 \end{bmatrix}$$

Having defined the row functions, the solved approximation problem can be transformed to the least squares method with

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{r_A}(v_{\theta,1}) \\ \vdots \\ \boldsymbol{r_A}(v_{\theta,N}) \end{bmatrix} = \begin{bmatrix} v_{\theta,1}^2 & v_{\theta,1} & 1 \\ \vdots & \vdots & \vdots \\ v_{\theta,N}^2 & v_{\theta,N} & 1 \end{bmatrix} \tag{4.64}$$

and

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{r_B}\left(\boldsymbol{\mu_{e,1}}, \boldsymbol{\Sigma_{e,1}}\right) \\ \vdots \\ \boldsymbol{r_B}\left(\boldsymbol{\mu_{e,N}}, \boldsymbol{\Sigma_{e,N}}\right) \end{bmatrix} = \begin{bmatrix} e_{x,1} & e_{y,1} & e_{\theta,1} & m_{1,1} & m_{2,1} & m_{3,1} & m_{4,1} & m_{5,1} & m_{6,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ e_{x,N} & e_{y,N} & e_{\theta,N} & m_{1,N} & m_{2,N} & m_{3,N} & m_{4,N} & m_{5,N} & m_{6,N} \end{bmatrix} \tag{4.65}$$

which will then return the matrix with resulting coefficients

$$\boldsymbol{X} = \begin{bmatrix} a_{e_x} & a_{e_y} & a_{e_\theta} & a_{m_1} & a_{m_2} & a_{m_3} & a_{m_4} & a_{m_5} & a_{m_6} \\ b_{e_x} & b_{e_y} & b_{e_\theta} & b_{m_1} & b_{m_2} & b_{m_3} & b_{m_4} & b_{m_5} & b_{m_6} \\ c_{e_x} & c_{e_y} & c_{e_\theta} & c_{m_1} & c_{m_2} & c_{m_3} & c_{m_4} & c_{m_5} & c_{m_6} \end{bmatrix} \tag{4.66}$$

and it is then also possible to compute the sum of 2-norms of the approximation errors, referred to as **residuals**,

$$\boldsymbol{e} = \begin{bmatrix} e_{e_x} & e_{e_y} & e_{e_\theta} & e_{m_1} & e_{m_2} & e_{m_3} & e_{m_4} & e_{m_5} & e_{m_6} \end{bmatrix} \tag{4.67}$$

expressing how accurate the resulting approximation is.

In the last step, it is necessary to consider that those results (the matrix $\boldsymbol{X}$) correspond to the approximation of the uncertainty after having driven a specific distance d', which is in this case $d' = 3\,\mathrm{m}$. Yet we need to obtain an approximation for the unit distance $d_0 = 1\,\mathrm{m}$, which could be used to compute an actual contribution to the uncertainty after one step of the update of the result of the EKF processing data of the robot's wheel encoders and IMU. It means we need to use those values to make a formula to be multiplied by the distance driven between 2 updates of the EKF. It is simple. We only divide the values of $\boldsymbol{X}$ by d' (and adjust the units of the values in the representation of $\boldsymbol{X}$), as it will be multiplied by a difference of distance, which is expressed in meters. Therefore the **approximated contribution to the error mean vector and the vectorized upper diagonal block of the symmetric covariance matrix** corresponding to the error mean vector can be expressed as

$$\Delta \begin{bmatrix} e_x & e_y & e_\theta & m_1 & m_2 & m_3 & m_4 & m_5 & m_6 \end{bmatrix} (d, v_x, v_\theta) \doteq \left(\frac{1}{d'} \boldsymbol{X}(v_x) \begin{bmatrix} v_\theta^2 d \\ v_\theta d \\ d \end{bmatrix} \right)^T , \tag{4.68}$$

where v_θ is the commanded angular velocity, v_x is the commanded forward velocity, $\boldsymbol{X}(v_x)$ is the matrix of coefficient expressed by (4.66) depending on the commanded v_x, d is the amount of driven distance between 2 consecutive updates of the EKF and d' is the distance driven to obtain the $\boldsymbol{X}(v_x)$.

Chapter 5

Implementation

5.1 The control of the robot's movement

5.1.1 The EKF-fused odometry

The odometry, whose data are published in the topic `/odometry/filtered` (with average rate of **50 Hz***), is a result of applying the **Extended Kalman Filter** (**EKF**) to data of the **encoder odometry** (read from the topic `/jackal_velocity_controler/odom` with average rate of **50 Hz***) and to data of the **IMU's gyroscope** (read from the topic `/imu/data` with average rate of **76 Hz***)[1]. It is maintained by the **robot_pose_ekf** ROS package (for details of the package please refer to [26]). The EKF is described in the Chapter 4, in the section 4.1 on page 25.

5.1.2 Jackal's driving controller and its optional setup parameters

Jackal's movement is driven from the `jackal_velocity_controller` node using the `diff_drive_controller` ROS package, as has been already mentioned, which is run by Jackal's **internal PC**. This controller package uses the ideal differential drive model described above for the approximation of the skid-steering model being driven.

[1]* Average rates of the published topics are computed by the `rostopic hz <topic name>` ROS Linux command

When it is launched, it loads its configuration from a local configuration file. There are several options by which the controller may be configured.

Configurable parameters of the `diff_drive_controller` ROS package

Here is the official list of all the configurable parameters with their data types, default values and explanation taken from [25], logically divided into groups of parameters and reordered:

Wheels and joints names:

1. **left_wheel** (string | string[...])
   - Left wheel joint name or list of joint names
2. **right_wheel** (string | string[...])
   - Right wheel joint name or list of joint names

Covariance matrices:

1. **pose_covariance_diagonal** (double[6])
   - Vectorized upper diagonal block of the covariance matrix for publishing of the pose of the odometry
2. **twist_covariance_diagonal** (double[6])
   - Vectorized upper diagonal block of the covariance matrix for publishing of the twist (basically a vector of velocities) of the odometry

Wheels and ICR settings:

1. **wheel_separation** (double)
   - The distance of the left and right wheel(s). The diff_drive_controller will attempt to read the value from the URDF if this parameter is not specified
2. **wheel_radius** (double)

- Radius of the wheels. It is expected they all have the same size. The diff_drive_controller will attempt to read the value from the URDF if this parameter is not specified.

3. **wheel_separation_multiplier** (double, default: 1.0)
    - Multiplier applied to the wheel separation parameter. This is used to account for a difference between the robot model and a real robot.
4. **wheel_radius_multiplier** (double, default: 1.0)
    - Multiplier applied to the wheel radius parameter. This is used to account for a difference between the robot model and a real robot.

Linear movement limits:

1. **linear/x/has_velocity_limits** (bool, default: false)
    - Whether the controller should limit linear speed or not.
2. **linear/x/max_velocity** (double)
    - Maximum linear velocity (in m/s)
3. **linear/x/min_velocity** (double)
    - Minimum linear velocity (in m/s). Setting this to 0.0 will disable backwards motion. When unspecified, **-max_velocity** is used.
4. **linear/x/has_acceleration_limits** (bool, default: false)
    - Whether the controller should limit linear acceleration or not.
5. **linear/x/max_acceleration** (double)
    - Maximum linear acceleration (in m/s^2)
6. **linear/x/min_acceleration** (double)
    - Minimum linear acceleration (in m/s^2). When unspecified, **-max_acceleration** is used.
7. **linear/x/has_jerk_limits** (bool, default: false)
    - Whether the controller should limit linear jerk or not.
8. **linear/x/max_jerk** (double)
    - Maximum linear jerk (in m/s^3).

Angular movement limits:

1. **angular/z/has_velocity_limits** (bool, default: false)
   - Whether the controller should limit angular velocity or not.
2. **angular/z/max_velocity** (double)
   - Maximum angular velocity (in rad/s)
3. **angular/z/min_velocity** (double)
   - Minimum angular velocity (in rad/s). Setting this to 0.0 will disable counter-clockwise rotation. When unspecified, -max_velocity is used.
4. **angular/z/has_acceleration_limits** (bool, default: false)
   - Whether the controller should limit angular acceleration or not.
5. **angular/z/max_acceleration** (double)
   - Maximum angular acceleration (in rad/s^2)
6. **angular/z/min_acceleration** (double)
   - Minimum angular acceleration (in rad/s^2). When unspecified, -max_acceleration is used.
7. **angular/z/has_jerk_limits** (bool, default: false)
   - Whether the controller should limit angular jerk or not.
8. **angular/z/max_jerk** (double)
   - Maximum angular jerk (in rad/s^3).

Publishing and subscribing settings.

1. **enable_odom_tf** (bool, default: true)
   - Publish to TF directly or not
2. **base_frame_id** (string, default: base_link)
   - Base frame_id, which is used to fill in the child_frame_id of the Odometry messages and TF.
3. **odom_frame_id** (string, default: "/odom")

- Name of frame to publish odometry in.

4. **publish_rate** (double, default: 50.0)
   - Frequency (in Hz) at which the odometry is published. Used for both tf and odom
5. **cmd_vel_timeout** (double, default: 0.5)
   - Allowed period (in s) allowed between two successive velocity commands. After this delay, a zero speed command will be sent to the wheels.
6. **publish_cmd** (bool, default: False)
   - Publish the velocity command to be executed. It is to monitor the effect of limiters on the controller input.
7. **allow_multiple_cmd_vel_publishers** (bool, default: False)
   - Setting this to true will allow more than one publisher on the input topic, `~/cmd_vel`. Setting this to false will cause the controller to brake if there is more than one publisher on `~/cmd_vel`.

Problems with the driving controller

There was once an attempt to limit the speeding-up acceleration of the robot, but we wanted the robot to keep slowing down (virtually braking) quickly enough not to collide with an obstacle. Therefore the **linear/x/max_acceleration** was decreased to $+5.0\ \mathrm{m/s^2}$, but the original **linear/x/min_acceleration** negative value was kept at its original negative value of $-20.0\ \mathrm{m/s^2}$. Afterwards the limit's effect has been tested in experiments.

The forward ride showed that the robot's acceleration has been truly limited while the slowing down was kept rapid as before, which was correct. But during effort to test the the robot's acceleration limit during the backward ride, it has shown that there is an **error in the driving controller's code**. Instead of the same slow acceleration and rapid deceleration, the robot did it vice versa. It accelerated rapidly and when there was the need to stop it, it decelerated slowly, which almost caused its collision with a cabinet.

Apparently, the driving controller does not distinguish whether the robot is currently riding forward or backward, when the controller should swap the limit constants. Instead of it the controller takes the limit constants as they are, which means that the upper and lower limit, respectively, don't mean the limit of acceleration and deceleration, respectively, but it only blindly limits the actual value of the derivative of the robot's forward speed.

There was an **attempt to correct the controller's code**, to recompile it and to integrate it to the robot's **jackal_velocity_controller** node. The code was corrected, but it could not be integrated to the robot's driving node, because it is secured by Clearpath and **integrated deeply in the internal PC**'s system as a secured daemon. Therefore the robot is still driven by the original controller with the conception of the acceleration limiting not distinguishing the acceleration and deceleration.

5.2 Libraries used in the scripts

The scripts are written in the Python language. Is reflects in the libraries used in them. The used libraries besides the standard python libraries are:

1. **Numpy**[2] - the famous Python library to perform algebraic computations,
2. **Matplotlib**[3] - the famous Python library to plot graphs,
3. **Rospy + ROS message descriptions**[4] - the ROS interface library for Python along with the specifications of the ROS messages,
4. **Pyrosbag**[5] - a Python class enabling the script to play a **rosbag** (= an archive containing chronologically recorded ROS messages from topics selected by a user) *and*
5. **KBHIT**[6] - a Python class implementing the keyboard interrupt manager **KBHIT**, enabling to check for and read pending keyboard keys pressed by a user

[2]Available at https://numpy.org/install/
[3]Available at https://matplotlib.org/users/installing.html
[4]Available at http://wiki.ros.org/rospy
[5]Available at https://pypi.org/project/pyrosbag/
[6]Available at https://gist.github.com/michelbl/efda48b19d3e587685e3441a74457024

5.3 The script for controlling the robot's driving and generally its measurement

As a part of this work, a measurement script, represented by the file **`print_odom.py`**, has been programmed. This script enables to command the robot to drive automatically and to do automatically other things, such as commanding the laser tracker to measure the retroreflectors or capturing and saving periodically images from the camera after each n driven meters. The measurement script also limits the accelerations and jerk, which would be caused by the commanded linear and angular velocity. Having been initialized, the script virtually resets the pose of the EKF-fused odometry by publishing a null pose to the topic **`/set_pose`**.

During the measurement, another ROS node, written by colleague Libor Wagner, is used. It is a ROS node maintaining communication with the laser tracker and providing a service for measuring an accurate position of a retroreflector with Leica by passing it the retroreflector's approximate coordinates in Leica's frame as an argument.

This script has been written entirely by the author of this work except for 1 item. The function `measure_with_leica()`, enabling the script to measure the positions of the wanted triplet of retroreflectors, has been written by colleague David Štych, during experiments with the tracking system, and then slightly extended by the author of this work so, that it can save data for the 2nd experiment.

In order to be able to use the laser tracker, it has to be calibrated. The only thing, which the laser tracker needs for its calibration, is a solidly placed retroreflector, which may be also one of the retroreflectors placed on the robot. Then the **`at4xx-cli`** program, which has been also written by Libor Wagner, is called with the arguments **`--init`** and **`--ip <the laser tracker's IP address>`**. Then the laser tracker begins to automatically calibrate itself.

5.3.1 Command line arguments of the script

The measurement script can be run with the following arguments:

`--trajectory=<line|circle|planned|vive-leica-calibration|none>` Specification of how the following arguments should be taken. Meaning of the values of this argument are:

`line` means that the **`--vel-lin`** and **`--vel-ang`** arguments are commanded.

`circle` means that the commanded linear and angular velocities are computed from the arguments **`--radius`** and **`--angle-rad`**.

`planned` means that the robot is controlled by an external source of commanded velocities and this script is used only for i. e. taking data.

`vive-leica-calibration` means that the robot is driven manually by the gamepad in order to obtain data for the 2nd experiment.

`none` (default) means that the robot is normally driven manually by the gamepad.

`--vel-lin=<m_s>` Specification of the commanded linear velocity - default 0.5 (m/s).

`--vel-ang=<rad_s>` Specification of the commanded angular velocity - default 0 (rad/s).

`--distance=<meters>` Specification of the distance to drive and stop after only when the `line` trajectory is specified.

`--radius=<meters>` Specification of the radius of the circular trajectory only when the `circle` trajectory is specified.

`--angle-rad=<radians>` Specification of the angle to drive and stop after only when the `circle` trajectory is specified.

`--acc-coef=<n>` Specification of the acceleration and deceleration when starting and stopping automatically

`--stop-period-m=<meters>` Specification of the distance after which to automatically stop the robot.

`--stop-period-rad=<radians>` Specification of the angle in radians after which to automatically stop when the `circle` trajectory is specified.

`--stop-period-deg=<degrees>` Specification of the angle in degrees after which to automatically stop when the `circle` trajectory is specified.

`--photo-period-m=<meters>` Specification of the distance after which to automatically capture an image from the camera.

`--photo-period-rad=<radians>` Specification of the angle in radians after which to automatically capture an image from the camera when the `circle` trajectory is specified.

`--photo-period-deg=<degrees>` Specification of the angle in degrees after which to automatically capture an image from the camera when the `circle` trajectory is specified.

`--auto-reaccelerate=<true|false>` Specification of whether to automatically reaccelerate after having been automatically stopped.

`--images-pgm` Specification of whether to save captured images in the **pgm** format.

`--images-ppm` Specification of whether to save captured images in the **ppm** format.

`--images-npy` Specification of whether to save the ROS messages containing the captured images as a packed Numpy array file (**.npy**).

`--images-folder=<path>` Specification of the folder where to save the captured images.

`--use-vive` Specification of whether to record also poses of the tracking system.

`--vive-side=<left|right>` Specification of the side of the robot on which positions of the retroreflectors will be measured by the laser tracker.

`--no-graph` Specification of whether not to show the graphs of recorded poses after the measurement is done.

`--use-leica` Specification of whether to use the laser tracker to measure positions of the retroreflectors.

`--print-when-stopped` Specification of whether to mark moments when the robot stopped to the final graph.

`--auto-go-forward-and-backward-count=<n>` Specification of the count of loops to drive periodically forward and backward based on the above specified arguments.

`--measure-vive-topic=<topic_name>` Specification of the name of the topic from which to record data.

5.3.2 Commands of the gamepad

The controls of the gamepad used in this script are the following:

Left joystick Manual driving of the robot.

L2 Dead man switch for this script.

L1 Increased velocity of the manual driving.

$\times + \bigcirc$ Capture an image by the camera and save it.

$\times + \square$ Terminate the script.

$\times$ Decelerate and stop.

$\triangle$ (Re)accelerate and go.

$\wedge$ Start the movement forward.

$\vee$ Start the movement backward.

$\bigcirc$ Measure the positions of the retroreflectors - when the **`vive-leica-calibration`** trajectory is speified.

5.4 The script for evaluating the measured data

This script, represented by the file **`plot_path.py`**, generally uses the mathematics described in the chapter 4: Theory. It has the following 5 functions. Generally, if it is needed in the particular functionality it may study data from one or more of the topics mentioned in the Chapter 3, in the subsection 3.1.3 on page 18.

5.4.1 Evaluating the results of the 2nd experiment

This functionality is run as **`python plot_path.py calibrate_vive_leica <datafile_path>`**, where the **`datafile`** is the file saved after running the script **`print_odom.py`** with argument **`--trajectory=vive-leica-calibration`**, described above. It visualizes the error of the pose of the tracking system towards the referential data measured by the laser tracker in the way described in the Chapter 6, in the subsection 6.3.2 on page 72.

5.4.2 Plotting the data from the odometry and IMU topics for a repeated movement during the 3rd experiment

This functionality is run as **`python plot_path.py extract_rosbag_data [--save] [--no-animation] <baglist_path>`**. There the optional argument **`--save`** determines, whether also data of the IMU have to be visualized. This function is trying to detect a change of the direction (forward/backward) of the robot's movement only by observing the velocities. Whenever the change of the direction is detected in data of one of the observed odometries, an inverse transform corresponding to the pose of the odometry is computed, which will be further applied to the poses obtained from the particular odometry in order to virtually reset it to a null pose when the change of the direction is detected. This is done due to the fact that during the experiments, this work is trying to study the data of the odometries as if the robot repeatedly started from the origin, in order to estimate the probabilistic model of the uncertainty.

5.4.3 Extraction of the data from baglists (lists of rosbags)

This functionality is run as **`python plot_path.py extract_rosbag_data [--save] [--no-animation] <baglist_path>`**. There the optional arguments are:

`--save` Determines that the user wants to save the recorded data of the odometries for their further visualization.

`--no-animation` Determines that the user does not want to visualize the data during their extraction.

The script extracts data from all the **rosbags** in the corresponding **baglist** (see the description below). It first **saves the currently received set of positions of the retroreflectors measured by the laser tracker, as well as the actual final poses of the odometries**. Then it waits for the next measured positions of the retroreflectors. At the time of receiving them, the script saves the poses obtained from the data of the studied odometries along with the positions of the retroreflectors measured by the laser tracker. When the script ends, it informs the user in the standard output about the location, where it saved the extracted data as a Numpy packed object (**`*.npy`**).

Description of a baglist

A **baglist** is a text file containing information about list of input bagfiles. It has the following format:

- The first row contains 2 items separated by a space:
  1. `LEFT` or `RIGHT` denoting the side of the robot containing the measured retroreflectors *and*
  2. a title of the baglist, preferably in quotes
- The following rows have again a structure of 2 items separated by a space:
  1. name of the rosbag file *and*
  2. count of wanted successfully finished loops
- The rows may be commented out by prepending a hashtag (`#`) sign to it.

5.4.4 Computing and approximating the resulting mean error vectors and covariance matrices from the extracted data

This functionality is run in 2 steps. First, the mean error vectors and covariance matrices are computed and the results are visualized gradually for all the extracted data contained in **`datafile`**s, which were saved during the extraction from the considered partial experiments of the **3rd experiment** in the previous step, by running **`python plot_path.py evaluate_extracted_data [--no-vive] [--save-statistics] <datafile_path>`**. There the optional arguments are:

`--no-vive` Determines that the extracted data of the tracking system are ignored and are not processed.

`--save-statistics` Determines that the user wants to save the mean error vector and the covariance matrix for further processing in the next step. Therefore this argument **has to be passed**.

This informs the user, where it saved the resulting statistics during each run for each of the saved **`datafile`**s extracted from the recorded partial experiments. The mean error vectors and covariance matrices are computed in the way described in the Chapter 4, in the subsection 4.6.1 on page 40.

Finally, the user obtains the final quadratic approximation of the dependence of the mean error vector and error covariance matrix on the commanded angular velocity, split for the positive and for the negative forward velocity, by running **`python plot_path.py evaluate_statistics <list of output files from the previous step>`**. The resulting matrices of coefficients of the quadratic approximation are printed to the standard output. They are computed in the way described in the Chapter 4, in the subsection 4.6.2 on page 42.

5.4.5 Replotting the saved data of odometries from the topics

This functionality is run as **`python plot_path.py replot_saved_odometries [--next] [--no-xy] <datafile_path>`**. There the optional arguments are:

`--next` Determines that the user wants to automatically show the secondarily plot graphs of angles, velocities and accelerations.

`--no-xy` Determines that the user does not want to visualize the xy-components of data of the odometries and just wants to show the secondarily plot graphs of angles, velocities and accelerations.

It just loads the plot data saved during the extraction step and visualizes them in the corresponding graphs.

Chapter 6

Experiments

6.1 The reference coordinate frames considered in the experiments

The diagram of referential coordinate frames, transforms between them and positions of retroreflectors towards the referential frames, considered in the experiments, can be seen in the figure 6.1. The referential coordinate frames are all orthonormal Cartesian coordinate frames and are denoted as follows:

W is the world coordinate frame,

L is the Cartesian coordinate frame of the laser tracker,

V is the coordinate frame of the tracking system (V for Vive),

T is the local coordinate frame of the HTC Vive Tracker attached to the robot,

F is the coordinate frame of the EKF-fused odometry), **f** (the local coordinate frame of the robot in the EKF-fused odometry,

R is the coordinate frame of the raw encoder odometry *and*

r is the local coordinate frame of the robot in the raw encoder odometry.

In the following experiments, the major studied transforms from the point of view of the accuracy of their sources are

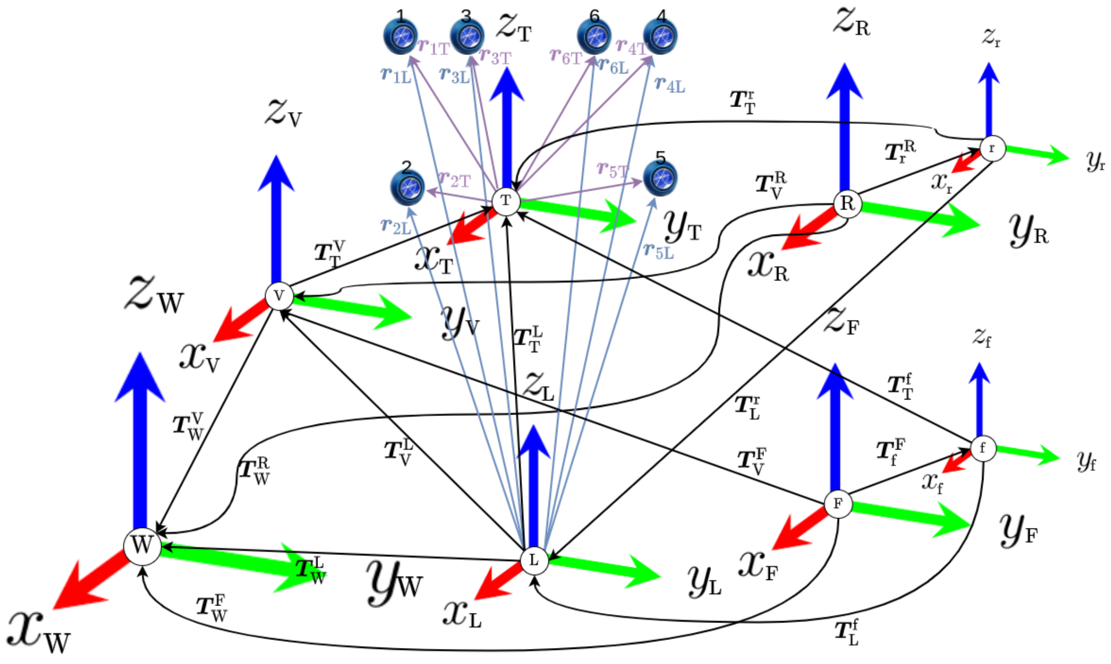

1
3
z_{T}
6
4
$\boldsymbol{r}_{1\mathrm{T}}$
$\boldsymbol{r}_{3\mathrm{T}}$
$\boldsymbol{r}_{6\mathrm{T}}$
$\boldsymbol{r}_{4\mathrm{T}}$
$\boldsymbol{r}_{1\mathrm{L}}$
$\boldsymbol{r}_{3\mathrm{L}}$
$\boldsymbol{r}_{6\mathrm{L}}$
$\boldsymbol{r}_{4\mathrm{L}}$
z_{R}
z_{r}
z_{V}
$\boldsymbol{T}_{\mathrm{T}}^{\mathrm{r}}$
$\boldsymbol{T}_{\mathrm{r}}^{\mathrm{R}}$
r
y_{r}
x_{r}
2
$\boldsymbol{r}_{2\mathrm{T}}$
$\boldsymbol{r}_{5\mathrm{T}}$
5
T
$\boldsymbol{T}_{\mathrm{V}}^{\mathrm{R}}$
R
y_{R}
x_{R}
$\boldsymbol{r}_{2\mathrm{L}}$
$\boldsymbol{r}_{5\mathrm{L}}$
y_{T}
$\boldsymbol{T}_{\mathrm{T}}^{\mathrm{V}}$
x_{T}
z_{W}
V
y_{V}
x_{V}
$\boldsymbol{T}_{\mathrm{T}}^{\mathrm{L}}$
z_{F}
z_{f}
$\boldsymbol{T}_{\mathrm{T}}^{\mathrm{f}}$
z_{L}
$\boldsymbol{T}_{\mathrm{L}}^{\mathrm{r}}$
$\boldsymbol{T}_{\mathrm{W}}^{\mathrm{V}}$
$\boldsymbol{T}_{\mathrm{V}}^{\mathrm{L}}$
$\boldsymbol{T}_{\mathrm{W}}^{\mathrm{R}}$
$\boldsymbol{T}_{\mathrm{V}}^{\mathrm{F}}$
$\boldsymbol{T}_{\mathrm{f}}^{\mathrm{F}}$
f
y_{f}
x_{f}
F
y_{F}
W
L
$\boldsymbol{T}_{\mathrm{W}}^{\mathrm{L}}$
y_{W}
x_{F}
y_{L}
x_{W}
$\boldsymbol{T}_{\mathrm{W}}^{\mathrm{F}}$
x_{L}
$\boldsymbol{T}_{\mathrm{L}}^{\mathrm{f}}$

Figure 6.1: Diagram of referential coordinate frames, transforms between them and positions of retroreflectors towards the referential frames, considered in the experiments. The referential coordinate frames are all orthonormal Cartesian coordinate frames and are denoted as: **W** (the world coordinate frame), **L** (Cartesian coordinate frame of the laser tracker), **V** (the coordinate frame of the tracking system (V for Vive)), **T** (the local coordinate frame of the HTC Vive Tracker attached to the robot), **F** (the coordinate frame of the EKF-fused odometry), **f** (the local coordinate frame of the robot in the EKF-fused odometry), **R** (the coordinate frame of the raw encoder odometry) and **r** (the local coordinate frame of the robot in the raw encoder odometry). The poses of the retroreflectors are expressed by the following vectors. For the i-th retroreflector ($i \in \{1, \dots, 6\}$), the vector $\boldsymbol{r}_{i\mathrm{L}}$ denotes its position in the coordinate frame **L** (of the laser tracker), which is measured by the laser tracker, and the vector $\boldsymbol{r}_{i\mathrm{T}}$ denotes its position in the coordinate frame **T** (of the tracker attached to the robot), which is constant and known from a manual measurement.

$\boldsymbol{T}_{\mathrm{r}}^{\mathrm{R}}$ obtained from the pose of the raw encoder odometry

$\boldsymbol{T}_{\mathrm{f}}^{\mathrm{F}}$ obtained from the pose of the EKF-fused odometry

$\boldsymbol{T}_{\mathrm{T}}^{\mathrm{V}}$ obtained from the pose published by the tracking system

The poses of the retroreflectors are expressed by the following vectors. For the i-th retroreflector ($i \in \{1, \dots, 6\}$), the vector $\boldsymbol{r}_{i\mathrm{L}}$ denotes its position in the coordinate frame **L** (of the laser tracker), which is measured by the laser tracker, and the vector $\boldsymbol{r}_{i\mathrm{T}}$ denotes its position in the coordinate frame **T** (of the tracker attached to the robot), which is constant and known from a prior manual measurement.

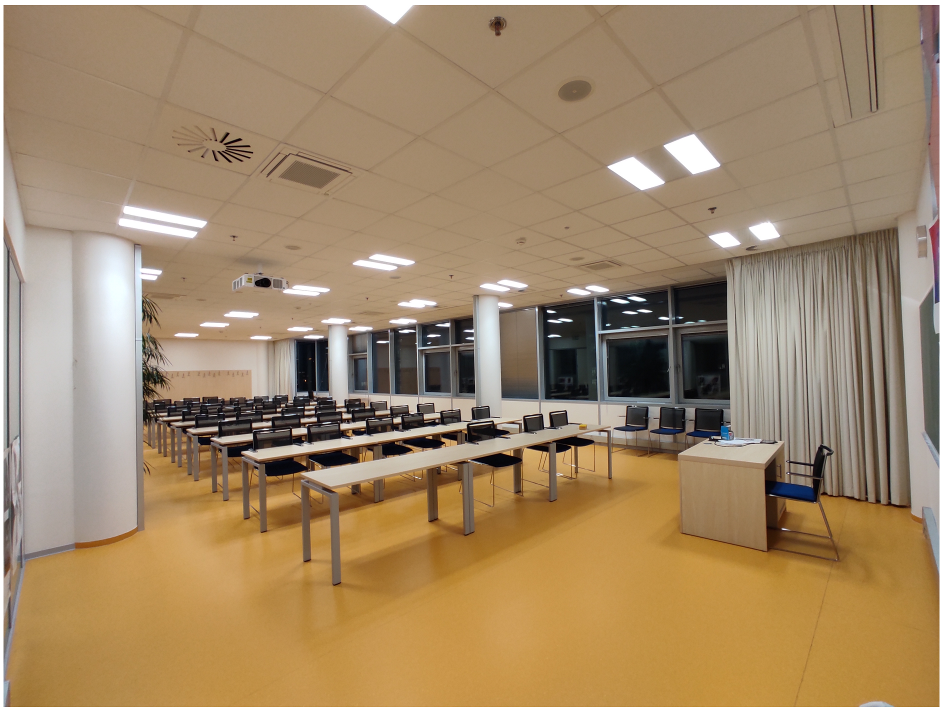
6.2 Observing the characteristics of the proprioceptive sensors of the robot

The robot was driven around a lecture hall (room B-670 on the 6th floor of CIIRC's building B, floor plan of which can be seen in the figures 6.2a through 6.3b) in closed loops. It was commanded to depart from an accurately defined pose, to drive the planned amount of loops without returning to the originating pose and to eventually return back to that originating pose. The robot set out from the spot under the sink next to the room's main door, facing the wall with its bottom, and was manually controlled by the gamepad to drive around the room on the following trajectories:

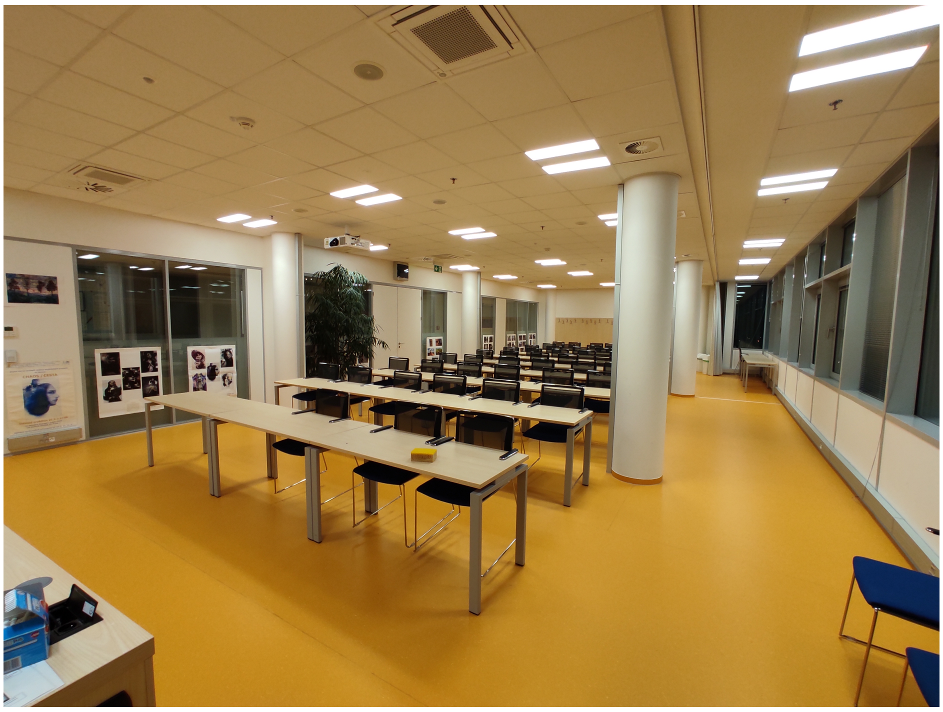
1. **One round** around the room driven in the way mentioned above
2. **Five rounds** around the room driven in the way mentioned above

During this experiment, the following data were recorded and then visualized, observed and evaluated:

1. raw encoder odometry (denoted by $\boldsymbol{T}_{\mathrm{r}}^{\mathrm{R}}$)
2. prefiltered IMU data
   - The **accelerometer** data **double integrated** for computing the **position**
   - The **gyroscope** data **integrated** for computing the angle $\boldsymbol{\theta}$
3. filtered odometry as EKF-fused encoder odometry with IMU data (denoted by $\boldsymbol{T}_{\mathrm{f}}^{\mathrm{F}}$)

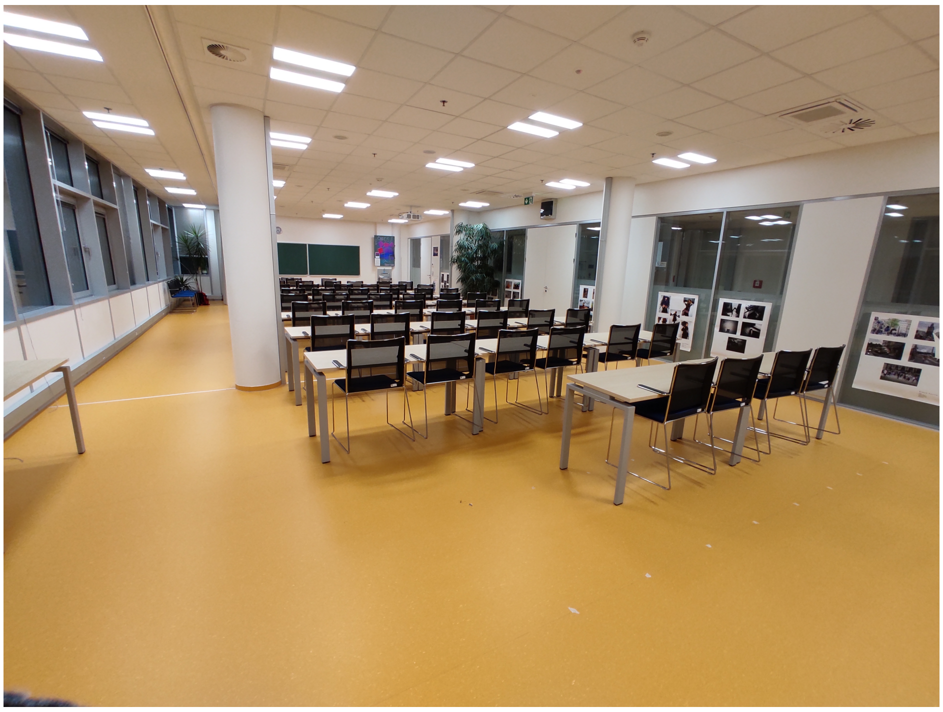


(a) : Front left view

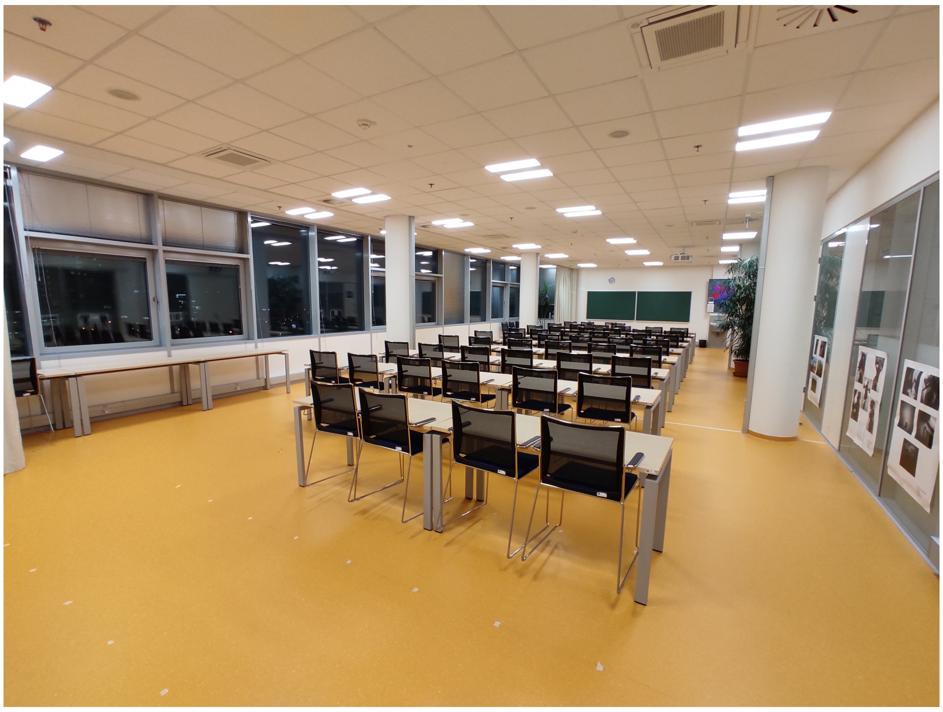


(b) : Front right view

Figure 6.2: Lecture hall B-670 at CIIRC - front wiews



(a) : Bottom left view



(b) : Bottom right view

Figure 6.3: Lecture hall B670 at CIIRC - bottom views

6.2.1 Results of the experiment

1 loop around the B-670

Having driven 1 round around the room, the recorded data of the odometries and integrated IMU data can be seen the graphs in the figures 6.4, 6.5 and 6.6.

It is apparent from the plots of data seen in the figures 6.4 through 6.9 when having driven 1 loop, that:

1. The **raw encoder odometry** keeps the information about **driven distance** relatively accurately, but not the information about the robot's orientation.
2. The **IMU integrated data (including single- and double-integrated accelerations and single-integrated angular velocities obtained from the IMU)** keeps the information about **the robot's orientation** relatively accurately (the only differences are caused by the fact that at some time intervals, the angles of IMU and the EKF-fused odometry are **wrapped to $\pm\pi$ differently**), but not the information about the robot's position.
3. The **EKF-fused odometry** keeps the information about both the **driven distance** and **the robot's orientation** relatively accurately.

It implies that the EKF-fused odometry apparently counts with the **distance data** from the **encoder odometry** and with the **orientation data** from the **IMU**.

It is also apparent from the graphs in the figures 6.4, 6.5 and 6.7 through 6.9 that the **IMU** has **biased accelerations**. It is probably caused by a **slight tilt of the IMU device** from the ideal horizontal position. If we study the velocities and accelerations, graphs of which are shown in the figures 6.7 and 6.8, respectively, it appears that both the x and y components of the acceleration vector are biased, as the velocities, obtained by integrating the IMU accelerations, keep growing. In order to prove this claiming, the acceleration means were computed as

$$\overline{a} = \frac{1}{n}\sum_{i=1}^{n} a_i\,, \tag{6.1}$$

where a_i is the respective x- or y-component of the i-th processed item of acceleration vectors published by the IMU and n is the total count of those processed acceleration vectors. The results

$$\overline{a_x} \doteq 0.0693\ \mathrm{m/s^2}$$
$$\overline{a_y} \doteq 0.0565\ \mathrm{m/s^2}$$

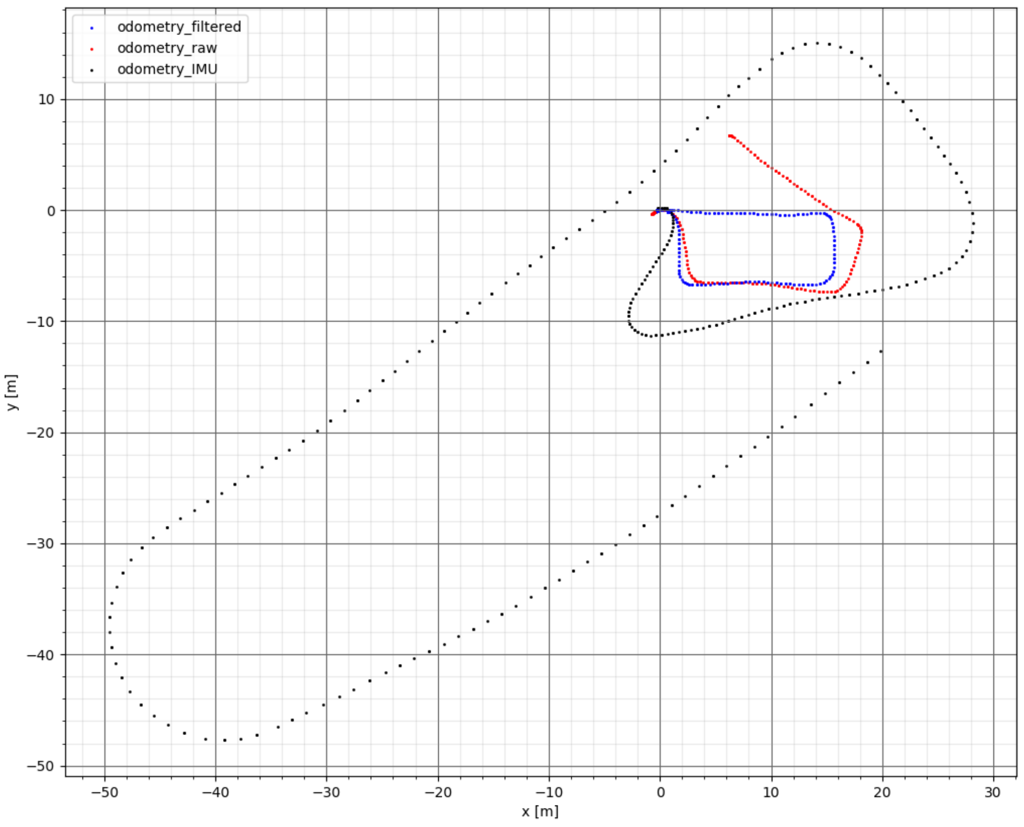


Figure 6.4: X-Y-data of odometries and IMU (1 round)

prove the claiming, as the non-zero mean accelerations mean that the accelerations will always have values such that they keep integrating, which results in permanent virtual acceleration seen in the trajectories taken from the double-integrated accelerations.

If we take another look at the graph of the velocities in the figure 6.7, we can see that the y-velocity of the integrated IMU accelerations is non-zero, whereas the y-velocities of the raw and filtered odometry are always zero. It explains the phenomenon that the path of the double-integrated IMU accelerations, shown in the figure 6.4, is significantly rotated in comparison with the EKF-fused path. It is so because the non-zero values of the y-components of the accelerations sensed by the IMU integrate into the non-zero lateral velocities and thus manifest in the tilt of the estimated trajectory against the real trajectory. In addition, since the accelerations are in the robot's local coordinate system, the local estimated trajectory contributions from the double-integrated IMU accelerations are always rotated by the actual estimated angle θ.

As it is apparent from the graph in the figure 6.7 that both the raw encoder odometry and the filtered odometry do not consider the y-component of the velocity. Eliminating the

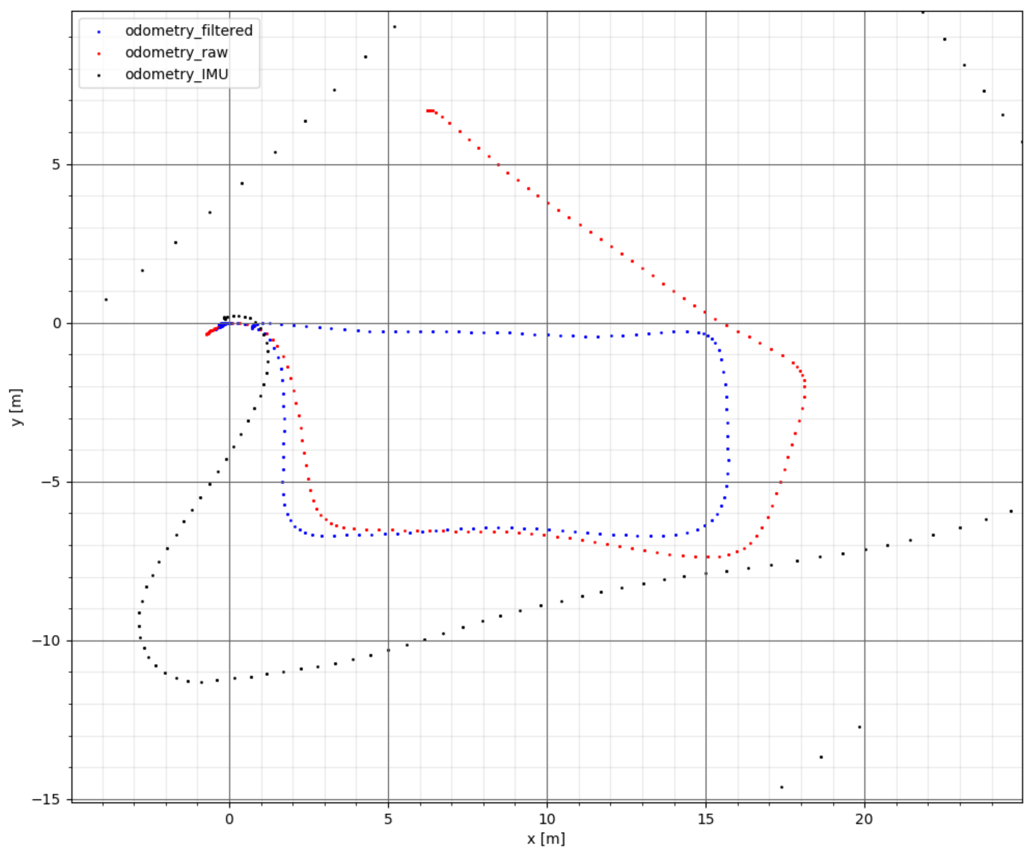


Figure 6.5: X-Y-data of odometries and IMU (1 round, detail)

y-component of the velocity of IMU, we get the following graph in the figure 6.9. There we can see that the orientation of the trajectory computed from the IMU data corresponds to the orientation of the trajectory obtained from the data of the filtered odometry and also apparently reflects robot's real trajectory.

5 loops around the B-670

Driving 5 rounds around the room has only proven the deductions. It can be seen in the graphs in the figures 6.10 through 6.13, that the EKF-fused odometry agrees

1. with the **raw encoder odometry** in **distances**, but **not angles**, and
2. with the **IMU-integrated-data odometry** in **angles (orientation)**, but **not distances**.

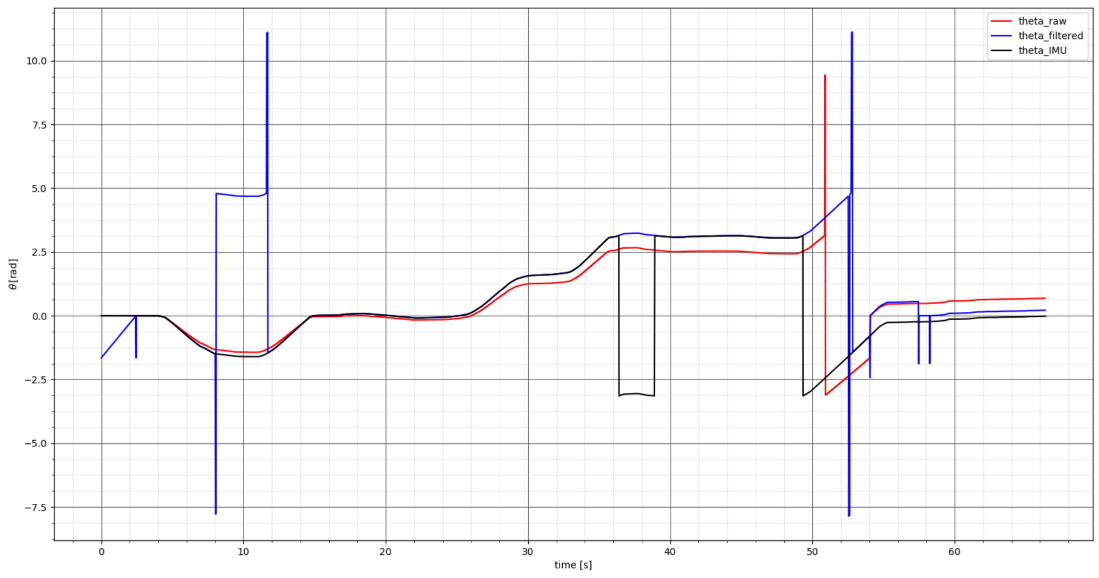


Figure 6.6: θ-t-data of odometries and IMU (1 round)

6.2.2 Implications

The results of this measurement lead to the following **implications about the sources of the uncertainty of the robot's odometry**:

1. The accuracy of the estimated **distance** depends **explicitly on the encoder odometry**.
2. The accuracy of the estimated **orientation** depends **explicitly on the IMU**.

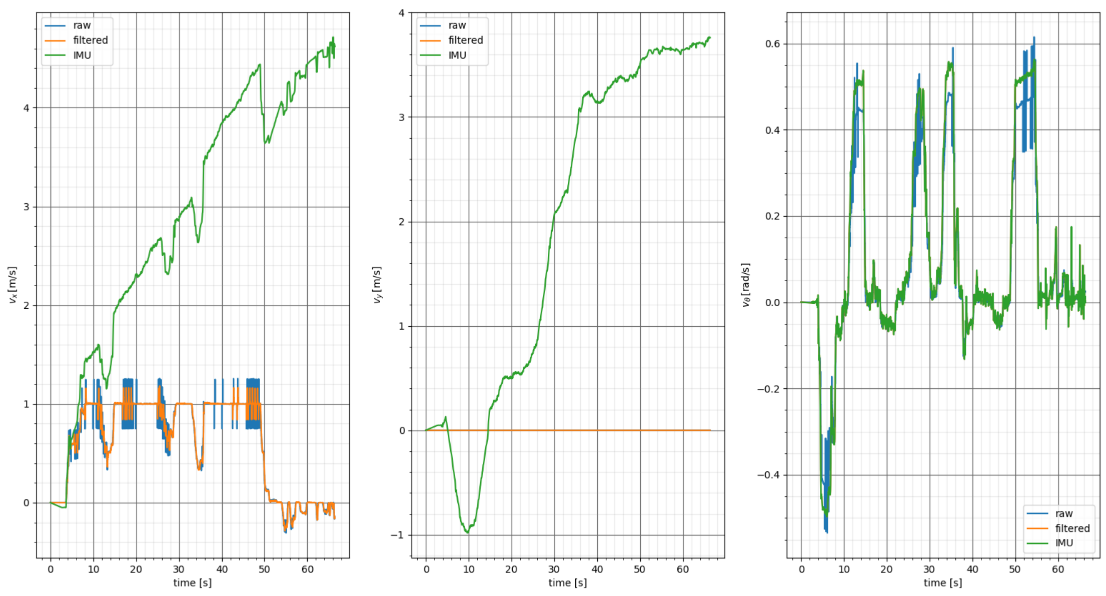


Figure 6.7: Robot's velocities from data of odometries and IMU (1 round)

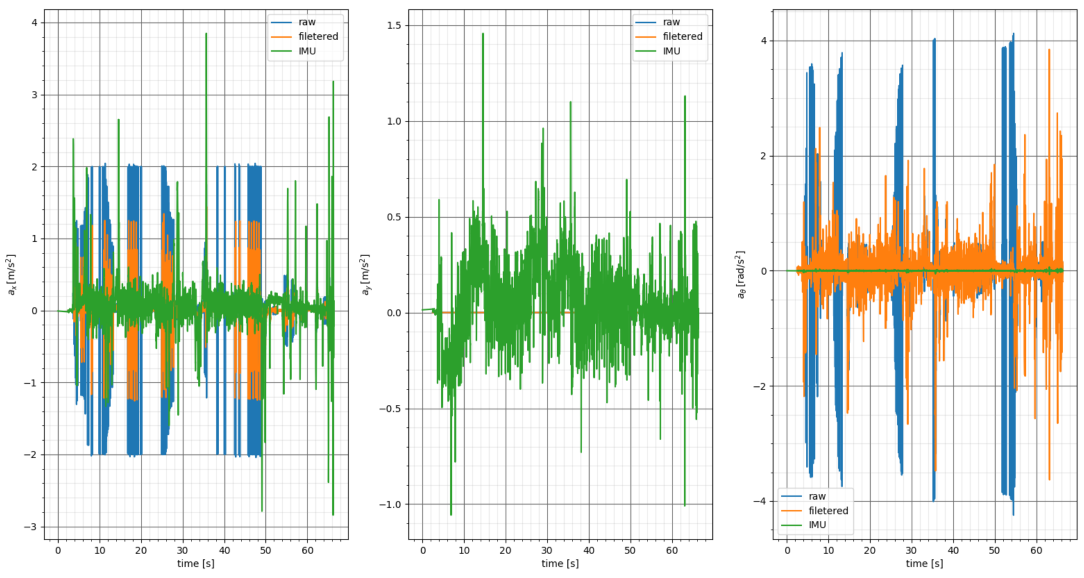


Figure 6.8: Accelerations from data of odometries and IMU (1 round)

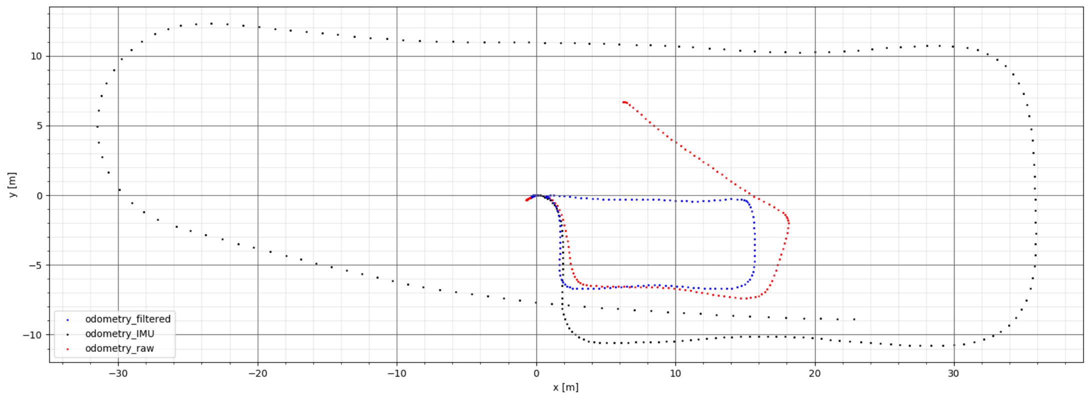


Figure 6.9: X-Y-data of odometries and IMU (eliminated y-velocities, 1 round)

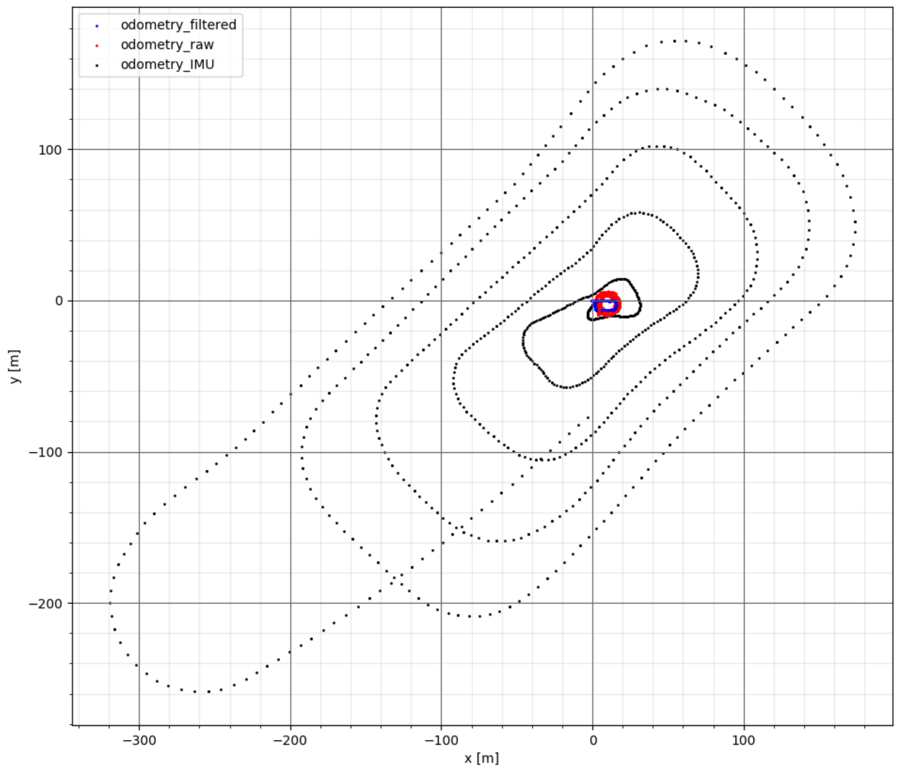


Figure 6.10: X-Y-data of odometries and IMU (5 rounds)

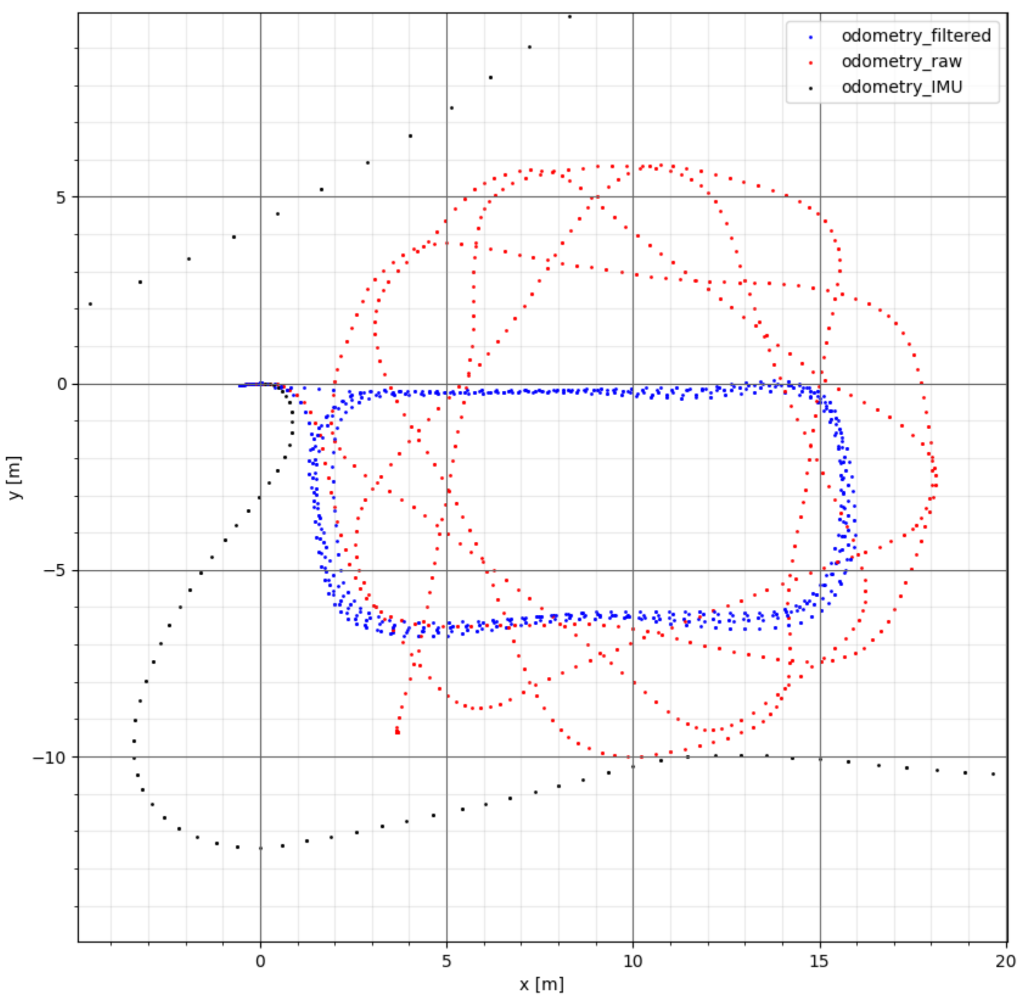


Figure 6.11: X-Y-data of odometries and IMU (5 rounds, detail)

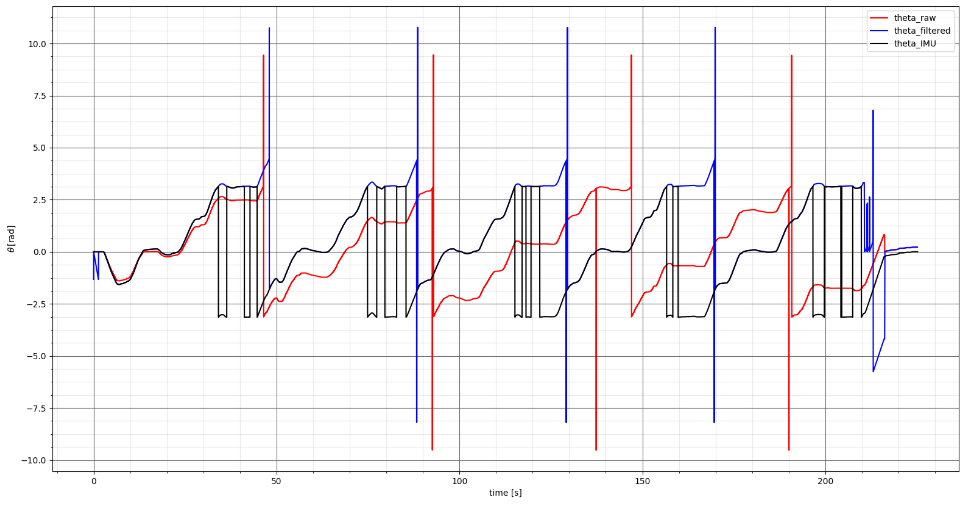


Figure 6.12: θ-t-data of odometries and IMU (5 rounds)

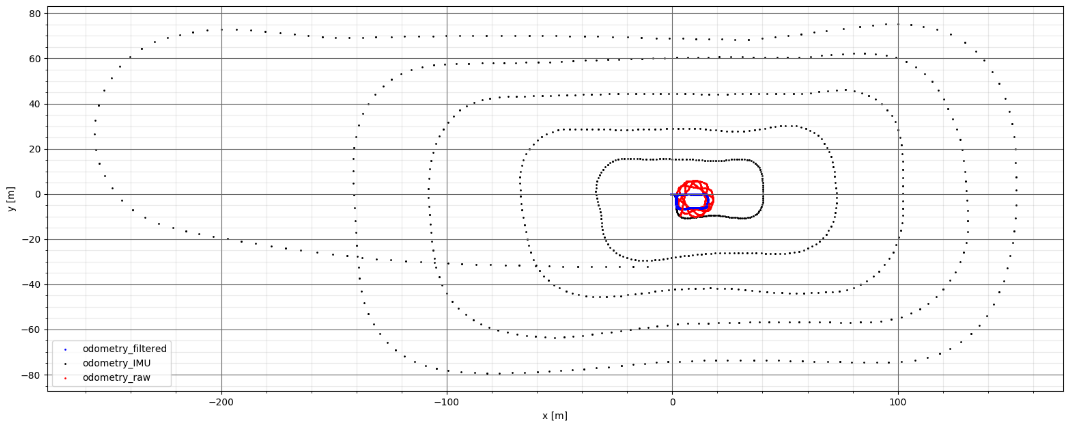


Figure 6.13: X-Y-data of odometries and IMU (eliminated y-velocities, 5 rounds)

6.3 Measuring the accuracy of the tracking system data

The goal of this experiment is to determine how accurately the tracking system's estimated pose, determining the transform $\boldsymbol{T}_{\mathrm{T}}^{\mathrm{V}}$, reflects the actual pose of the tracker attached to the robot, depending on the robot's distance from the pose where the transform $\widehat{\boldsymbol{T}}_{\mathrm{V}}^{\mathrm{L}}$ used in the guidance system has been estimated during the calibration. The calibration is further described back in the Chapter 4, in the section 4.5 on page 36. According to the result of this experiment, the processing of the next experiment will be adjusted.

6.3.1 Performing the experiment

The experiment was performed in a closed area such that the tracker system does not start halting, which happens when the robot gets to a position that is not well observable by the tracker system. For the experiment, a grid of 8 $\times$ 3 visual markers, placed in an area of 2 m $\times$ 3.5 m, which can be seen in the figure 6.14, has been prepared on the floor. This grid serves only for marking the approximate stopping points of the robot, being controlled manually with the gamepad during this experiment.

The principle of the experiment is the following. The robot is being manually controlled by the gamepad to stop at each of the grid's land markers, driving so that there is an effort to keep the direction of its local x-axis (its facing direction) parallel to the lines connecting the longer edge of the land markers grid. During each of the stops, the positions of the set of retroreflectors on the robot's right side are measured by the laser tracker, facing the mentioned longer edge perpendicularly so that it sees the robot from the side, as can be seen in the figure 6.15a.

The positions of the retroreflectors on the right side of the robot measured by the laser tracker, $\boldsymbol{r}_{1\mathrm{L}}$ through $\boldsymbol{r}_{3\mathrm{L}}$, are saved along with the actual corresponding set of estimated positions of the retrorflectors, $\widehat{\boldsymbol{r}}_{1\mathrm{L}}$ through $\widehat{\boldsymbol{r}}_{3\mathrm{L}}$, estimated by the previously calibrated guidance tool from the tracking system's **6DOF** pose, as has been described in the Chapter 4, in the section 4.5 on page 36. The triplet of the positions of the retroreflectors mounted on the side of the robot, which faces the laser tracker, estimated by the guidance tool system unequivocally determines the 6DOF pose of the tracker, mounted on the robot, estimated by the tracking system. The reference frames related to this experiment and relations among them are depicted in the diagram in the figure 4.4 on page 39. From there and from the description in the mentioned section 4.5 on page 36, it is apparent that we can evaluate the accuracy of the tracking system by comparing the corresponding triplets of the measured and of the estimated positions of the retroreflectors, $\boldsymbol{r}_{1\mathrm{L}}$ through $\boldsymbol{r}_{3\mathrm{L}}$ and $\widehat{\boldsymbol{r}}_{1\mathrm{L}}$ through $\widehat{\boldsymbol{r}}_{3\mathrm{L}}$, respectively. Doing the comparison, we may particularly study how the accuracy of the pose

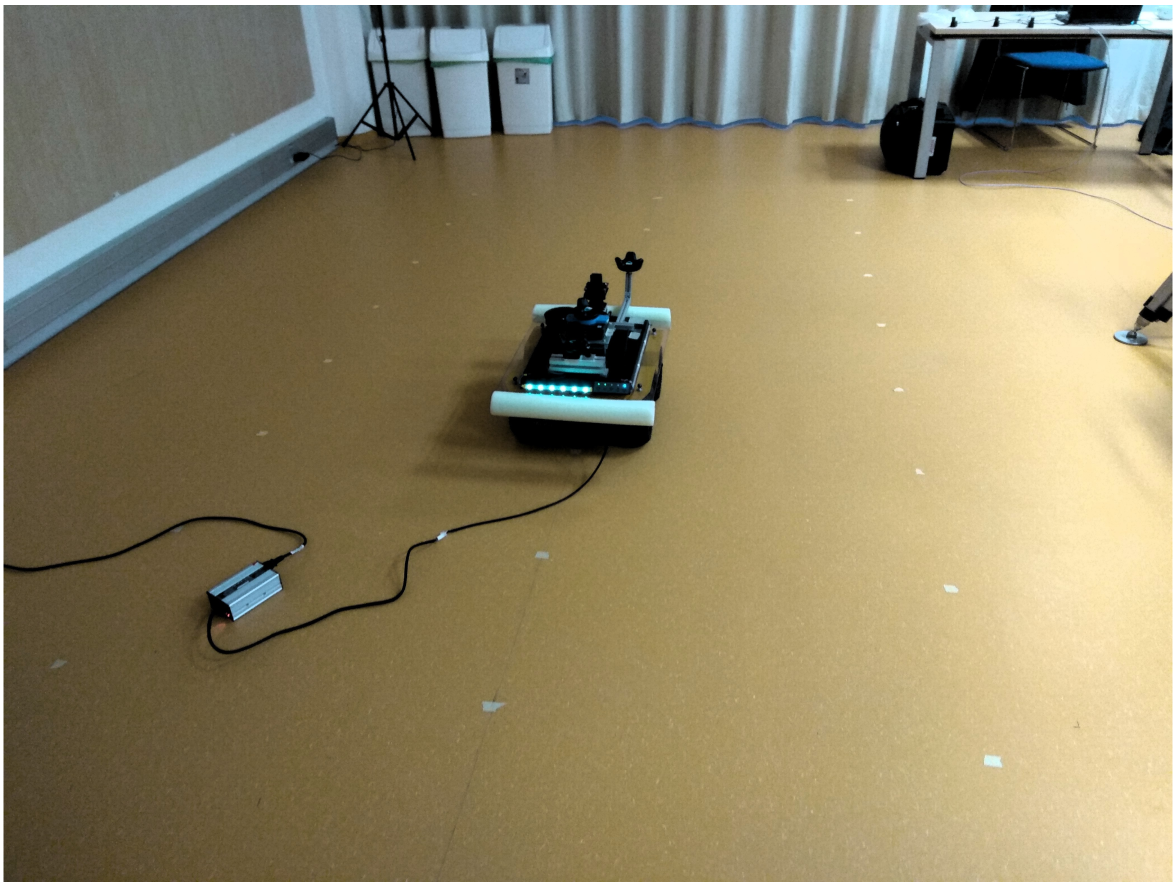


Figure 6.14: Land markers grid for approximate stopping

of the tracker, mounted on the robot, estimated by the tracking system evolves with increasing distance from the pose of the robot, where the guidance tool was calibrated.

During this experiment, the tracking system Lighthouses are placed and tilted so that their view ranges can include the entire measurement scene and the tracking system's pose can be thus estimated without problems, as can be seen in the figures 6.15a and 6.15b. As has been mentioned, the tracking system is sensitive to external radiation. Therefore the experiment was performed with drawn curtains and dimmed light.

6.3.2 Results of the experiment

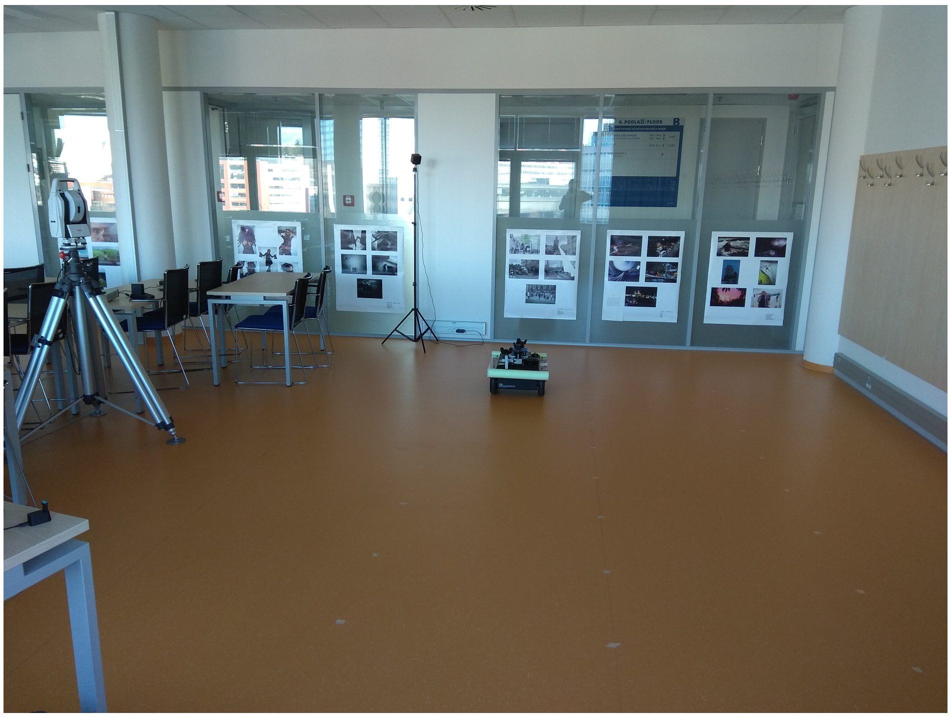
The characteristics of the $\boldsymbol{XY}$**-data** of centroids of position vectors $\boldsymbol{r}_{1\mathrm{L}}$ through $\boldsymbol{r}_{3\mathrm{L}}$ and $\widehat{\boldsymbol{r}}_{1\mathrm{L}}$ through $\widehat{\boldsymbol{r}}_{3\mathrm{L}}$, obtained at each of the landmarks of the grid described above, can be seen in the following graph in the figure 6.16. Those centroids $\boldsymbol{c}_i$ and $\widehat{\boldsymbol{c}}_i$, respectively, are computed using the equation (4.30) with substitutions $\boldsymbol{A}_3^i = \begin{bmatrix}\boldsymbol{r}_{1\mathrm{L}}^i & \boldsymbol{r}_{2\mathrm{L}}^i & \boldsymbol{r}_{3\mathrm{L}}^i\end{bmatrix}$ and $\boldsymbol{A}_3^i = \begin{bmatrix}\widehat{\boldsymbol{r}}_{1\mathrm{L}}^i & \widehat{\boldsymbol{r}}_{2\mathrm{L}}^i & \widehat{\boldsymbol{r}}_{3\mathrm{L}}^i\end{bmatrix}$, respectively, where i denotes the number of the measurement. There we may see

1. the **geometric centers $\boldsymbol{c}_i$ of triplets of retroreflector poses** $\boldsymbol{r}^i_{1\mathrm{L}}$ through $\boldsymbol{r}^i_{3\mathrm{L}}$ measured by **the laser tracker** as orange **“diamonds”**,
2. the **geometric centers $\widehat{\boldsymbol{c}}_i$ of triplets of retroreflector poses** $\widehat{\boldsymbol{r}}^i_{1\mathrm{L}}$ through $\widehat{\boldsymbol{r}}^i_{3\mathrm{L}}$ estimated from **the tracking system** by the guidance tool as blue **points** *and*
3. the **error vectors** from centroids $\boldsymbol{c}_i$ of $\boldsymbol{r}^i_{1\mathrm{L}}$ through $\boldsymbol{r}^i_{3\mathrm{L}}$ to centroids $\widehat{\boldsymbol{c}}_i$ of $\widehat{\boldsymbol{r}}^i_{1\mathrm{L}}$ through $\widehat{\boldsymbol{r}}^i_{3\mathrm{L}}$ having their **length multiplied by 25** (for better illustration), this computed as

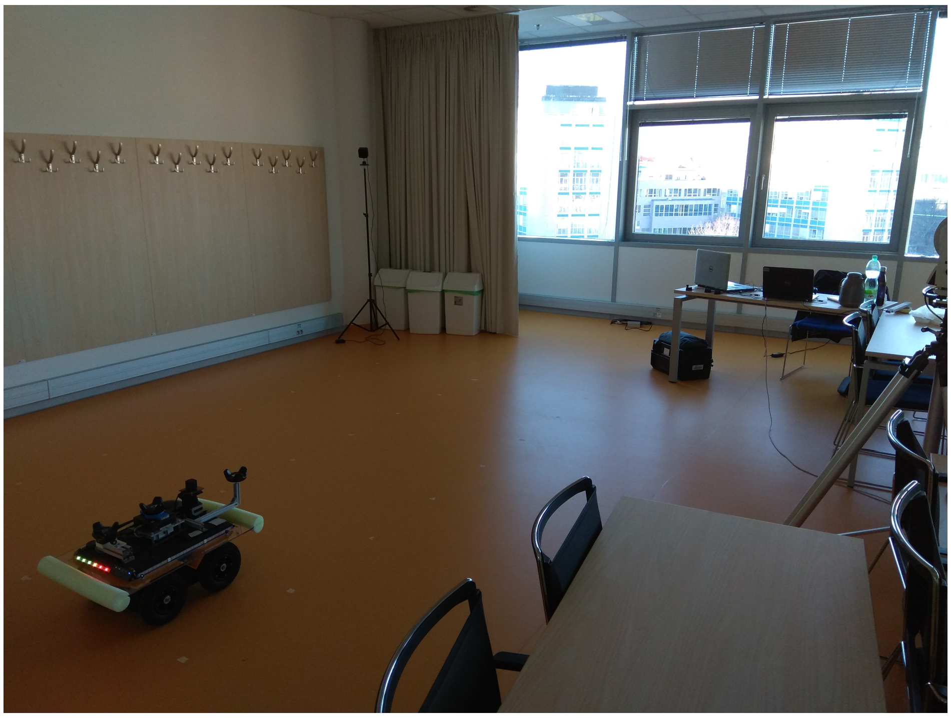
$$\boldsymbol{v}^i_e = 25\left(\widehat{\boldsymbol{c}}_i - \boldsymbol{c}_i\right) \tag{6.2}$$

as **black arrows**.

The resulting figure shows that the tracking system **odometry** is **not accurate enough to be used as a referential source of odometry**, yet that it is **accurate enough to be used for guiging the laser tracker to the approximate locations of retroreflectors**, as it can be seen from the figure that the largest error vector is only about 90 mm long, which is good enough for the laser tracker to adjust itself to the retroreflector (which is proven by the fact that the laser tracker was automatically guided by the tracking system-estimated retroreflector poses during this experiment). The tracking system odometry is thus enabling us to automatize the measurement process of the following experiment with use of the guidance tool.



(a) : the laser tracker and the first Lighthouse



(b) : The second Lighthouse

Figure 6.15: Setup of the measurement of the tracking system's linearity and accuracy

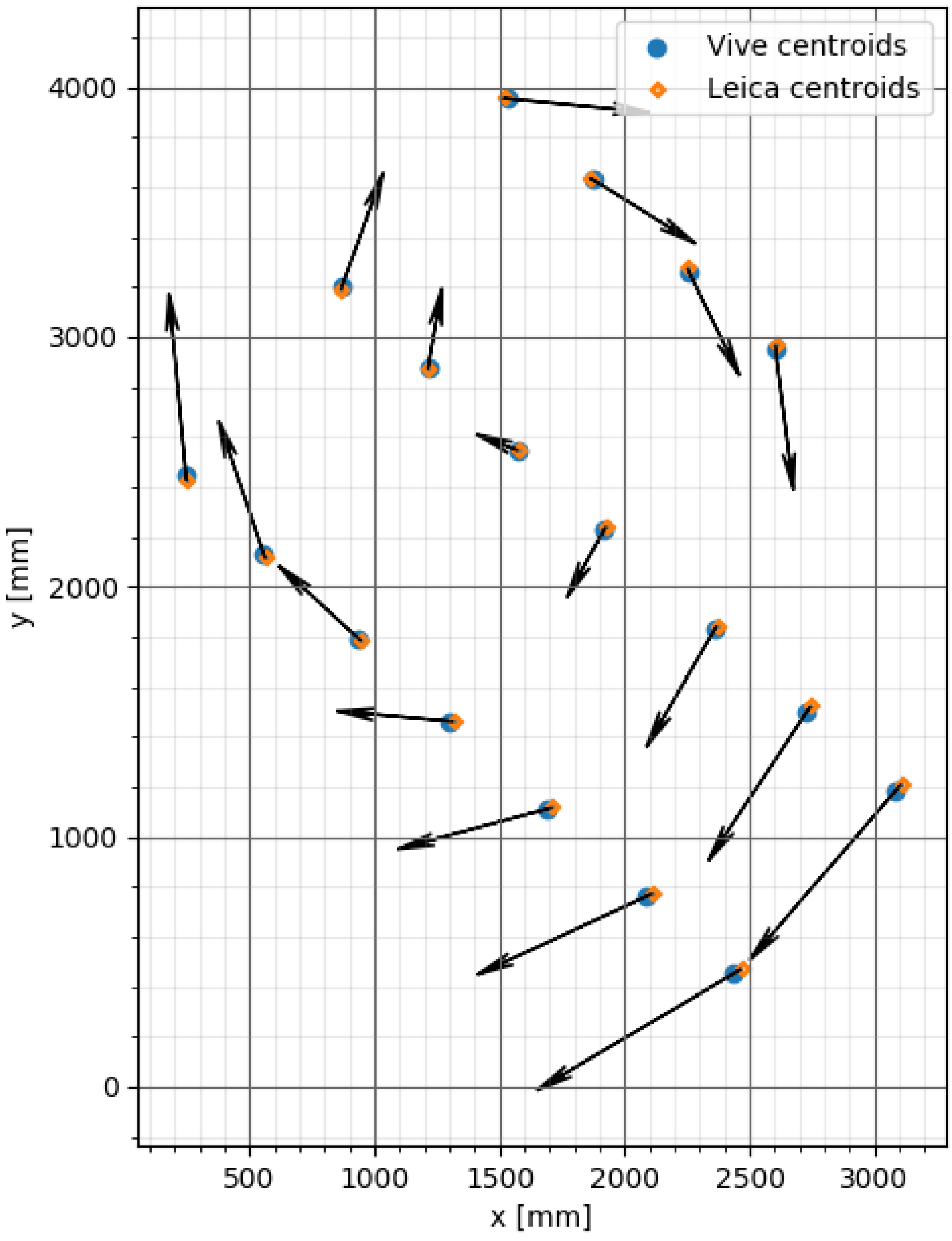


Figure 6.16: XY-characteristics of the tracking system centroids compared to the laser tracker centroids (arrows denote the error vectors mutliplied by 25) through the landmark grid

6.4 Measuring and estimating the uncertainty of the EKF-fused odometry of the robot

The goal of this experiment is to determine the probabilistic model of the uncertainty of the robot's odometry and to approximate the contribution to the ucertainty depending on the commanded liner and angular velocity and on the driven distance. The reference data, for comparing the odometry with which, were taken from the laser tracker measurement. The odometry is thus compared with the laser tracker data in order to model its uncertainty.

During this experiment, we need first to model the uncertainty of the EKF-filtered odometry by a mean error vector and an error covariance matrix, based on the experimental data and the commanded velocities and driven distances. Subsequently, we need to approximate the contribution to the mean error vector and the error covariance matrix based on the commanded forward and angular velocities and the increment of the driven distance. The mathematical background for this experiment is described in the Chapter 4, in the section 4.6 on page 40.

6.4.1 Performing the experiment

During this experiment, the robot was controlled by the measurement script to drive repeatedly forward and backward for at least 50 times the distance of **3 m** with the commanded forward velocities of **0.5 m/s** and **-0.5 m/s** and the following commanded angular velocities: **0 rad/s**, **0.1 rad/s**, **0.2 rad/s**, **-0.1 rad/s** and **-0.2 rad/s**.

Please note that the velocities, respectively the accelerations were limited during the acceleration and deceleration so that the robot smoothly accelerates and stops, doesn't skid and drives the distance of **3 m** as accurately as possible (at least what concerns the estimated distance from the EKF-fused odometry).

Before the first acceleration and after each following deceleration, the positions of the retroreflectors mounted on the robot's side facing the laser tracker were automatically measured by the laser tracker. the laser tracker faced roughly perpendicularly the baseline of the robot's movement during this experiment in order to maintain sufficiently small reflection angles bettween the laser tracker and the prisms of the retroreflectors. the laser tracker was commanded by the measurement script to measure the 3D point corresponding to the estimated positions of the retroreflectors, which were estimated by the node guiding the laser tracker according to the tracking system's pose.

We want to reset the pose of an odometry to the origin with null orientation every time the robot changes its direction of movement. Originally the measurement script reset the pose of an odometry when a change of the direction in that odometry was detected, but it has proved to be inaccurate. So the actual version resets poses of all odometries when a message containing a triplet of positions of the retroreflectors is received.

When the experiment was performed, data from various sources of information about the pose of the robot were recorded, including

1. **raw encoder odometry** determining the transform $\boldsymbol{T}_{\mathrm{r}}^{\mathrm{R}}$,
2. **EKF-fused odometry** determining the transform $\boldsymbol{T}_{\mathrm{f}}^{\mathrm{F}}$,
3. **the 6DOF pose of the tracking system** determining the transform $\boldsymbol{T}_{\mathrm{T}}^{\mathrm{V}}$,
4. **estimated positions of the retroreflectors** denoted by the vectors $\widehat{\boldsymbol{r}}_{1\mathrm{L}}$ through $\widehat{\boldsymbol{r}}_{6\mathrm{L}}$, published by the guidance tool *and*
5. **positions of retroreflectors** measured by **the laser tracker** denoted by the vectors $\boldsymbol{r}_{1\mathrm{L}}$ through $\boldsymbol{r}_{6\mathrm{L}}$.

As has been mentioned, the tracking system is sensitive to external radiation. Therefore the experiment was performed with drawn curtains and dimmed light.

6.4.2 Processing the data

Besides studying how the odometries and the robot's real pose evolve during the repeated movement, we want to study mainly the characteristics of relative errors of the data of the odometries during the robot's repeated movement as if the robot **set out each time from the origin, both during its forward and backward movement**. The data are first extracted from the recorded rosbags and subsequently, the extracted data are studied and evaluated.

The measured data data are extracted in the way described in the Chapter 5, in the subsection 5.4.3 on page 54. During the extraction, the **sets of positions of the retroreflectors measured by the laser tracker** and the **actual final poses of the odometries** are obtained.

These final poses and measured positions of the retroreflectors are processed and evaluated in the way described in the Chapter 4, in the section 4.6 on page 40.

6.4.3 Results of the partial experiments for the different commanded velocities

The visualization of results of this experiment can be seen in the graphs in the following figures 6.17 through 6.104, described in a more detailed way below.

The main goal of this experiment is to obtain a set of

- **means** of error vectors $\boldsymbol{\mu_e}(v_x, v_\theta, i_{IMU})$ ("**mean error vectors**") of the data of the filtered odometry towards the reference data measured by the laser tracker and
- their respective error **covariance matrices** $\boldsymbol{\Sigma_e}(v_x, v_\theta, i_{IMU})$,

where

- v_x is the commanded forward velocity (expressed in **m/s**),
- v_θ is the commanded angular velocity (expressed in **rad/s**) and
- i_{IMU} is a Boolean value of whether the IMU has been initialized properly or not (**true/false**)

The problem of proper or improper initialization of the IMU is described below. Briefly, sometime it happens that the IMU does not initialize properly during the booting of the robot, which has a significant impact on the accuracy of the filtered odometry, which was learned after the experiments were done from the recorded data. Let us then briefly study the results of each of the 5 kinds of trajectory (and their respective commanded velocities and states of the initialization of the IMU).

Driving straight 3 meters forward and backward at 0.5 m/s

This trajectory was driven in 53 successfully recorded loops. Particularly this experiment was performed with 2 several hours long breaks after first 13 and first 32 successfully measured entire loops, which showed one important source of errors. Depending on **whether the IMU initializes properly** during the robot's boot-up, the **filtered odometry is either significantly accurate or much less accurate**. The effect of the IMU's improper initialization can be seen in the graphs in the figures 6.17 and 6.18, respectively, where we

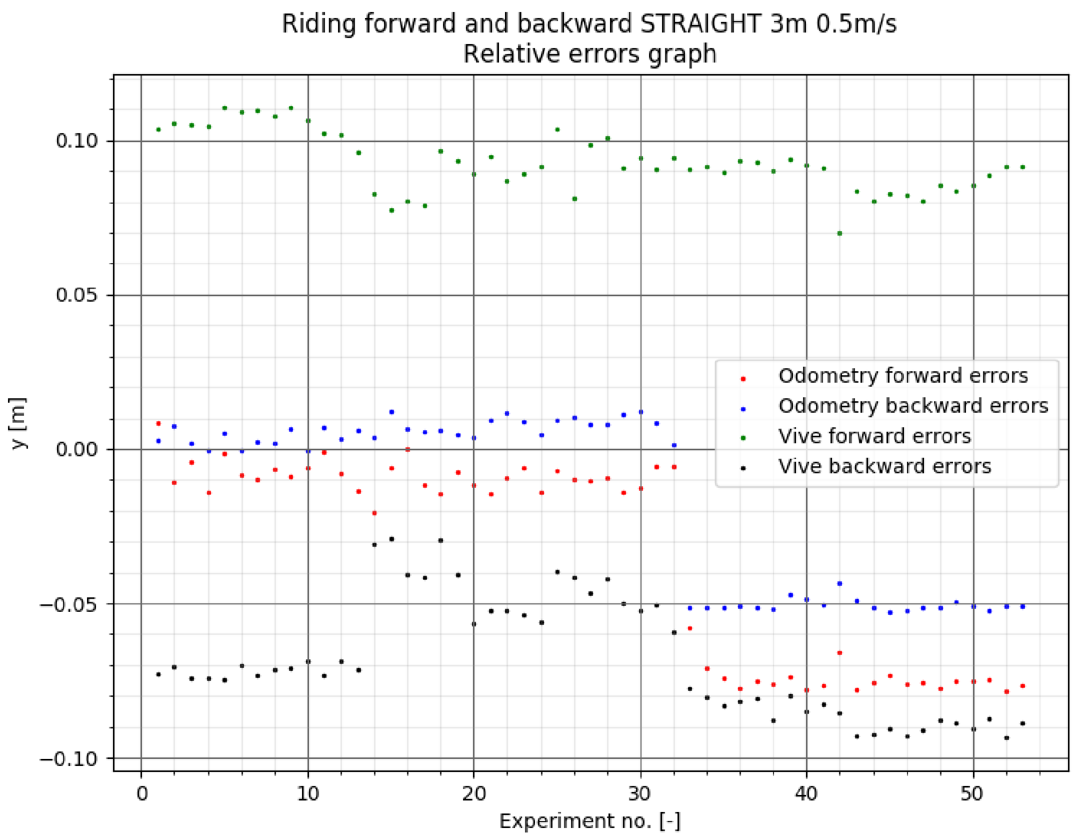


Figure 6.17: Dependency of the y-component of the local error of the odometry on the loop's order number (straight trajectory)

can see the y-components and θ-components, respectively, of the local error vector (in the final pose of the trajectory) depending on the loop's order number. The same goes for the accuracy of the tracking system's odometry according to the calibration accuracy. Whereas the **filtered odometry was more accurate during the first 32 loops** and then the improperly reinitialized IMU caused the increase in the values of the error vector components, **the tracking system's odometry was more accurate from the 14th to the 32nd loop**.

It leads to the implication that **the uncertainty of the filtered odometry has 2 distinct probabilistic distributions, depending on whether the IMU has been initialized properly or not**. It will have to be determined whether the IMU is initialized properly or not in the resulting improved localization algorithm. As will be seen further, it can be **determined from the behavior of the difference between the θ-components of the filtered odometry and of the raw encoder odometry**. If the difference is constantly null or a very small number, which does not change, as may be seen in the figure 6.33, it means that the IMU has not been initialized properly, whereas if the difference changes in time, as may be seen in the figure 6.25 (where the angles have even different signs), it means that the IMU has been initialized properly.

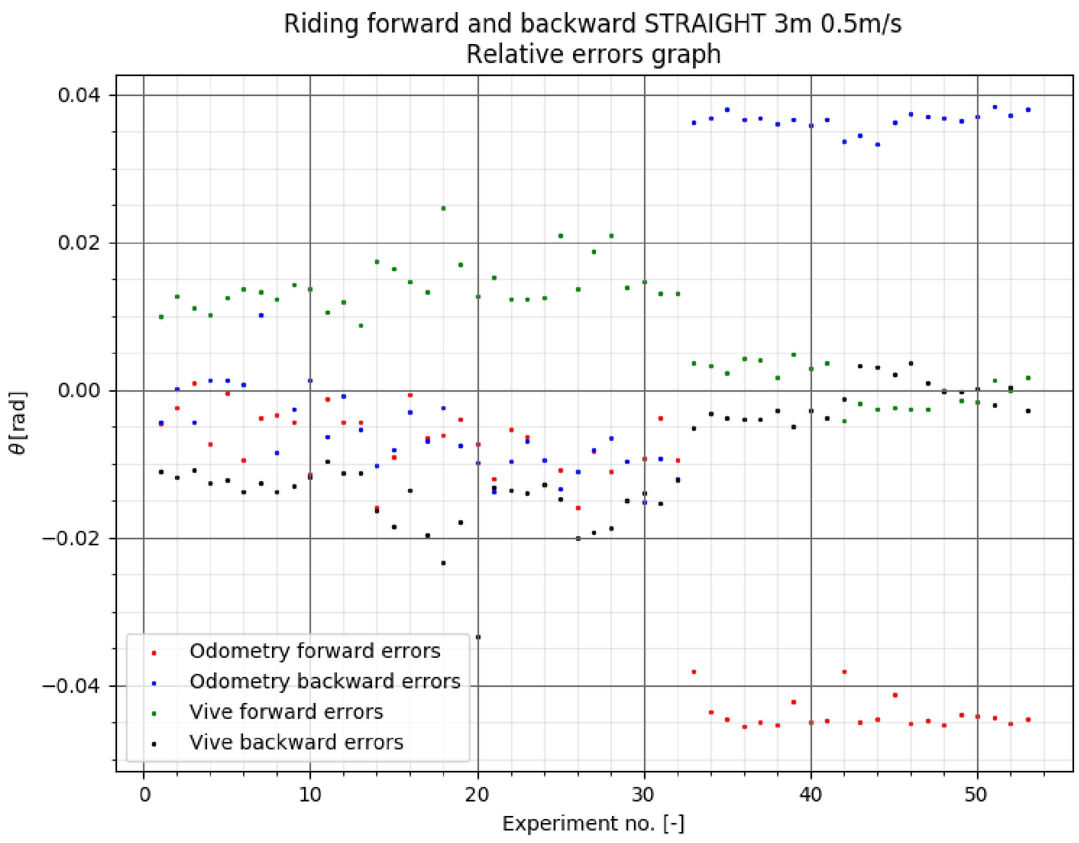


Figure 6.18: Dependency of the θ-component of the local error of the odometry on the loop's order number (straight trajectory)

The robot's trajectories and their final poses *were read from:*

1. the raw encoder odometry (only in some of the figures),
2. the filtered odometry *and*
3. the the tracking system odometry.

The **final poses** correspond to the poses at the moment right before the robot changes the direction of its drive from forward to backward or vice versa, which is assumed to right after having had its pose measured by the laser tracker. *The* ***plots of the robot's final poses****,* or of the entire trajectories (if specified in the captions), **adjusted always to the null position when changing its direction**, along with the ***respective graphs of errors****, can be seen*

1. in the figures 6.19[1] through 6.24 *for all the loops* (*including* ***both*** **first interval** with smaller error (**first 32 loops**) and **second interval** with larger error (**last 21 loops**)),
   - There is apparent that some trajectories of the filtered odometry (in the first interval) are outlying from the raw encoder odometry trajectories, whereas other trajectories of the filtered odometry (in the second interval) correlate with the raw encoder odometry trajectories, which means that no information from the IMU was then received.
2. in the figures 6.25 through 6.32 *with details for the* ***first*** *interval* of loops *and*
3. in the figures 6.33 through 6.40 *with details for the* ***second*** *interval* of loops.

The graphs show that **the robot keeps subtly steering left it the direction of its driving**, even though it is commanded to drive straight with null angular velocity. In reality it results in **permanently increasing tilt of the direction of the robot's straight trajectory counter-clockwise**, which may be seen in the graph in the figures 6.41 and 6.42 where the the poses are not adjusted to the null position when changing the robot's direction. Also final poses estimated from the laser tracker measurement appear in some of the graphs (marked in the legends of the particular graphs).

In the plots, it can be seen that even the raw encoder odometry notices slight steering leading to a zig-zag-like trajectory instead of straight trajectory. This means that it is sensed by the IRC's of the robot's motors, which leads to the conclusion that this phenomenon is not caused only by the robot's physics, yet it is caused also by a sort of **error in the**

[1]Please note that the outlier points in the graphs of trajectories are caused only by an imperfection in the data processing script.

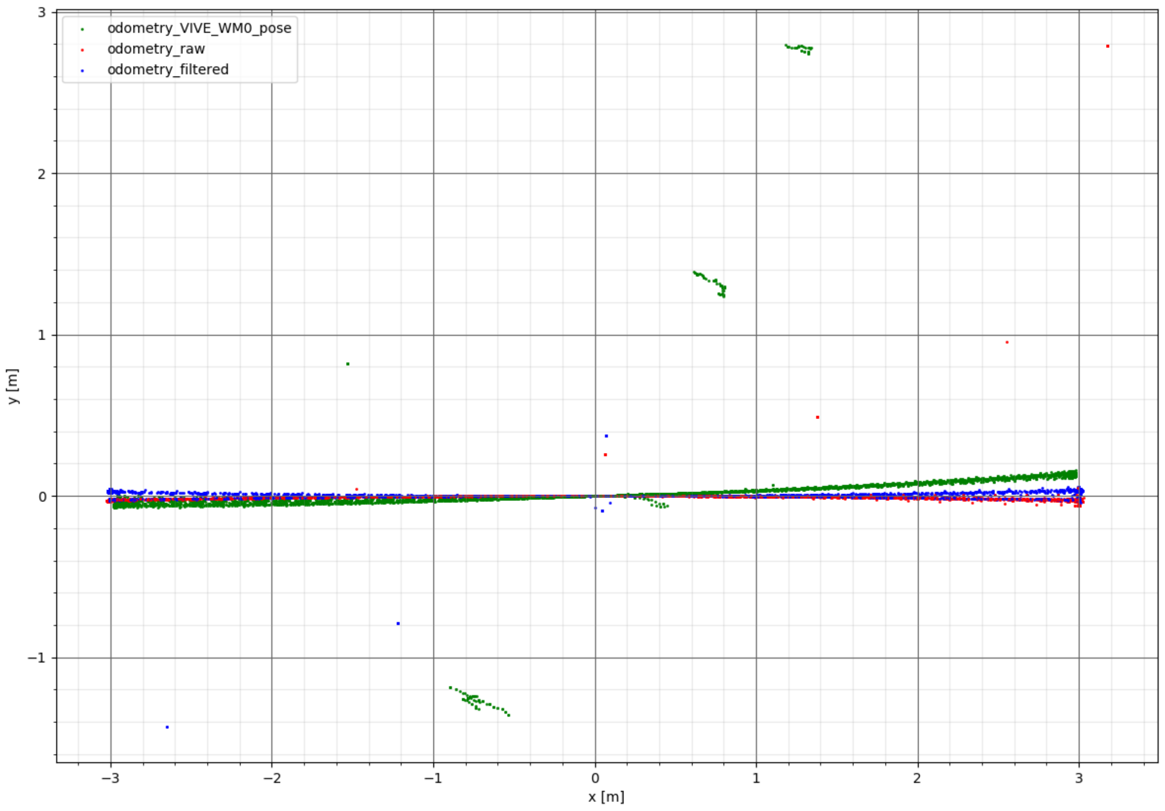


Figure 6.19: Plot of XY-data of the robot's adjusted trajectories when repeatedly driving straight forward and backward (both intervals; straight trajectory - null angular velocity)

Differential driving controller (described above in the subsection 5.1.2), which controls the robot's skid-steering driving. In addition, in case of properly initialized IMU, we may see in the graph of θ-components of the odometries in the figure 6.25 that the real value of θ estimated by the filtered odometry (of course with some uncertainty) has signs mostly opposite of the signs of θ-components of the raw encoder odometry.

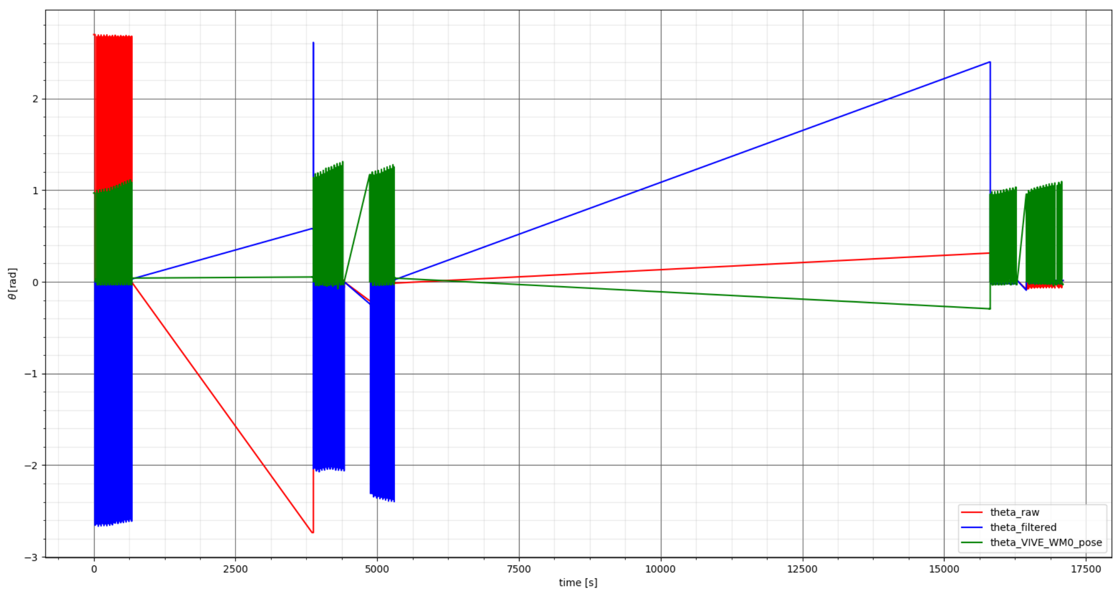


Figure 6.20: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving straight forward and backward (both intervals; not much legible, rather for the idea of breaks between the sub-intervals - detailed graphs will follow below)

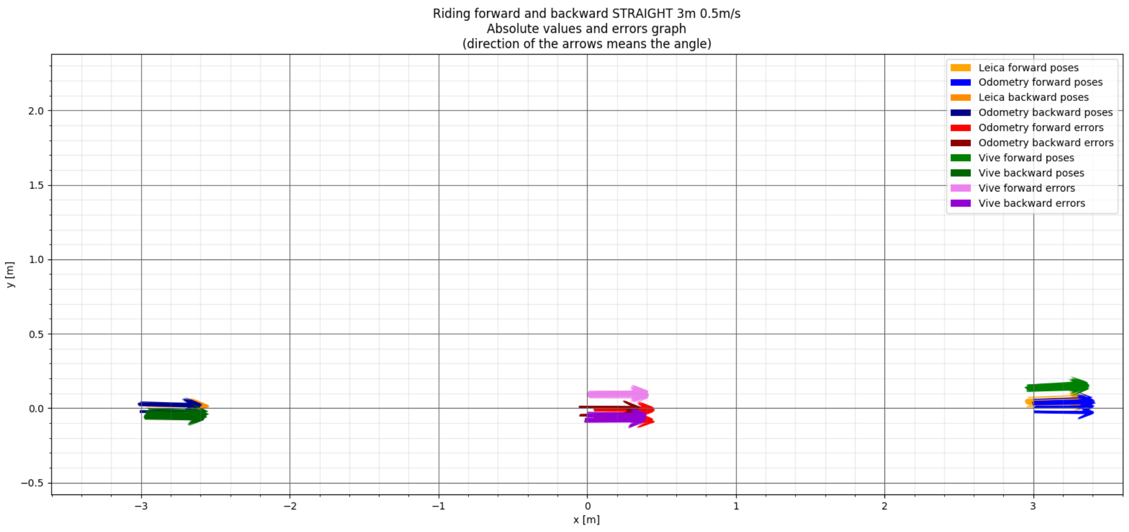


Figure 6.21: Plot of final poses and error vectors (towards the laser tracker final poses) of the robot's adjusted trajectories of odometries (both intervals; straight trajectory - null angular velocity)

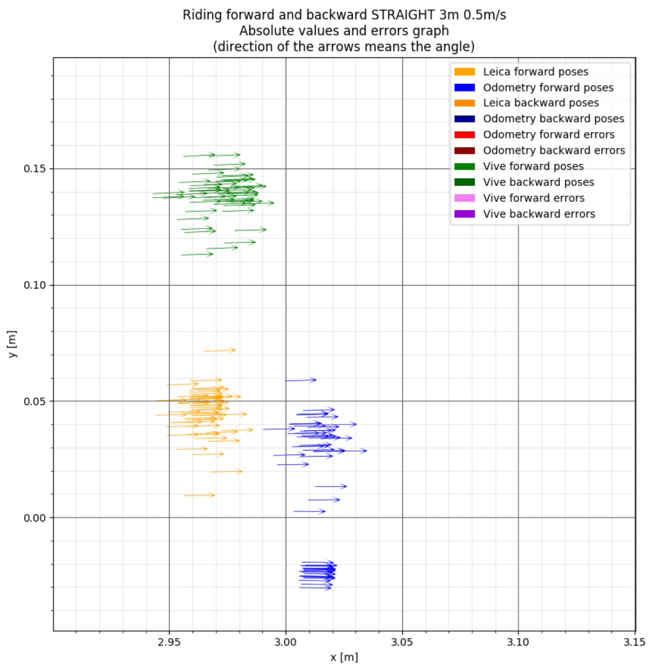


Figure 6.22: Plot of final poses of the robot's adjusted trajectories of odometries (both intervals; straight trajectory - null angular velocity, forward detail)

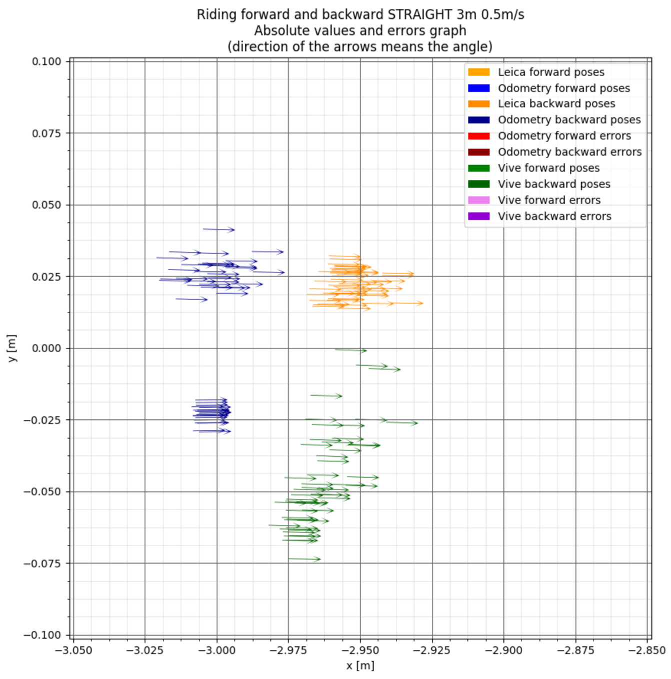


Figure 6.23: Plot of final poses of the robot's adjusted trajectories of odometries (both intervals; straight trajectory - null angular velocity, backward detail)

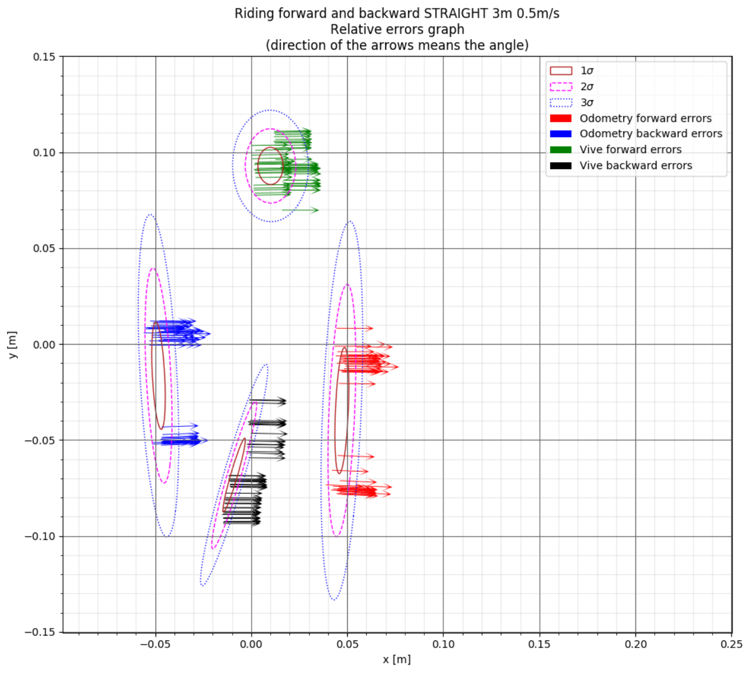


Figure 6.24: Plot of final error vectors of odometries towards the laser tracker reference data and their confidence ellipses (both intervals; straight trajectory - null angular velocity)

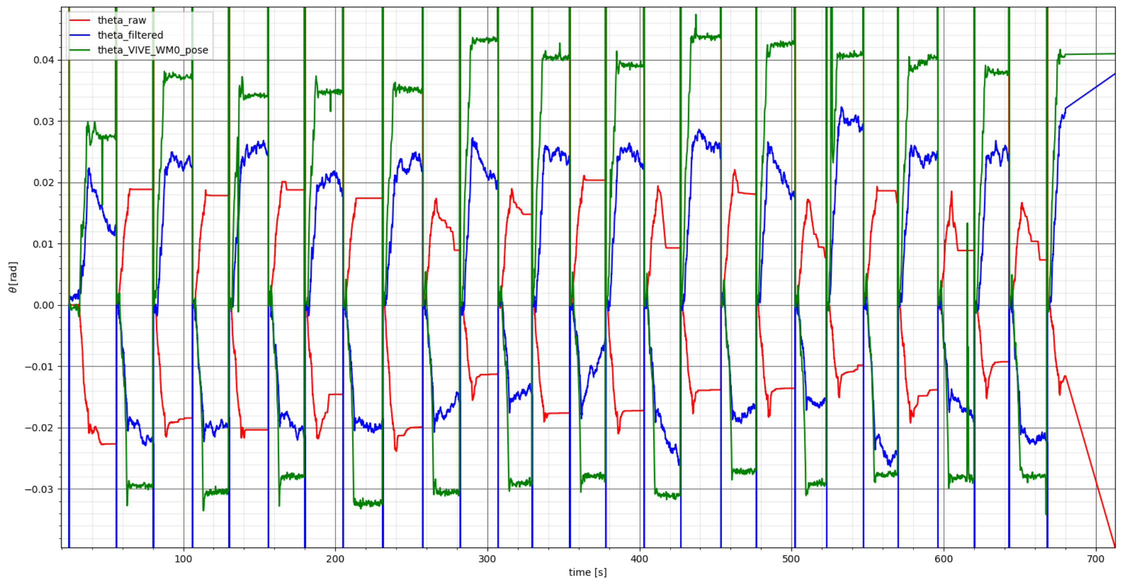


Figure 6.25: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving straight forward and backward (first part of the first interval; note that here the raw encoder odometry has even values with opposite signs)

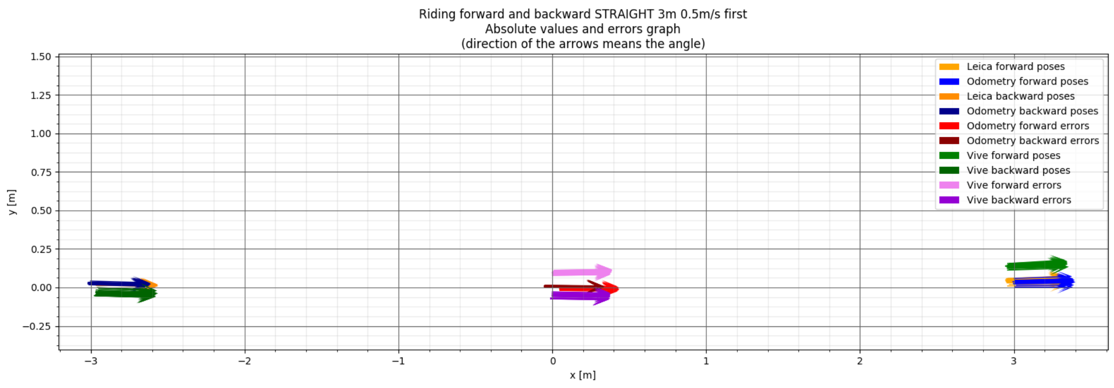


Figure 6.26: Plot of final poses and error vectors (towards the laser tracker final poses) of the robot's adjusted trajectories of odometries (first interval; straight trajectory - null angular velocity)

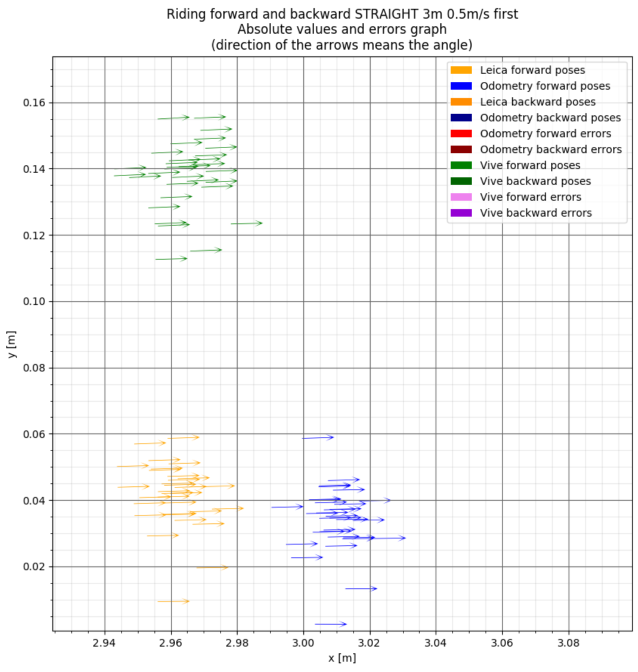


Figure 6.27: Plot of final poses of the robot's adjusted trajectories of odometries (first interval; straight trajectory - null angular velocity, forward detail)

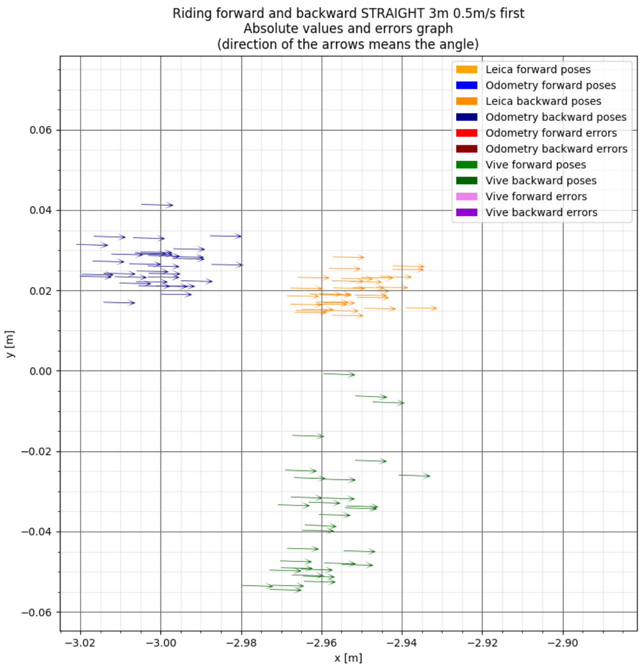


Figure 6.28: Plot of final poses of the robot's adjusted trajectories of odometries (first interval; straight trajectory - null angular velocity, backward detail)

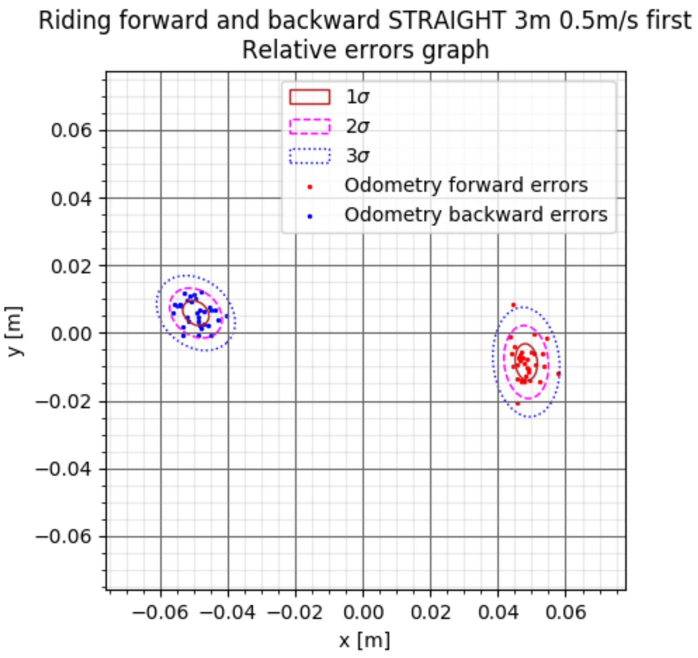


Figure 6.29: Plot of x and y components of relative errors of the final poses of the robot's adjusted trajectories of odometries (first interval; straight trajectory - null angular velocity)

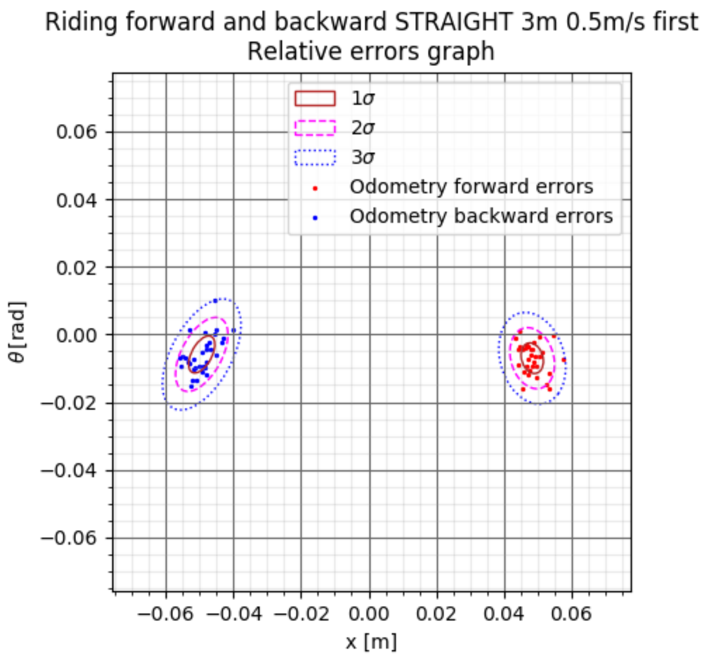


Figure 6.30: Plot of x and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (first interval; straight trajectory - null angular velocity)

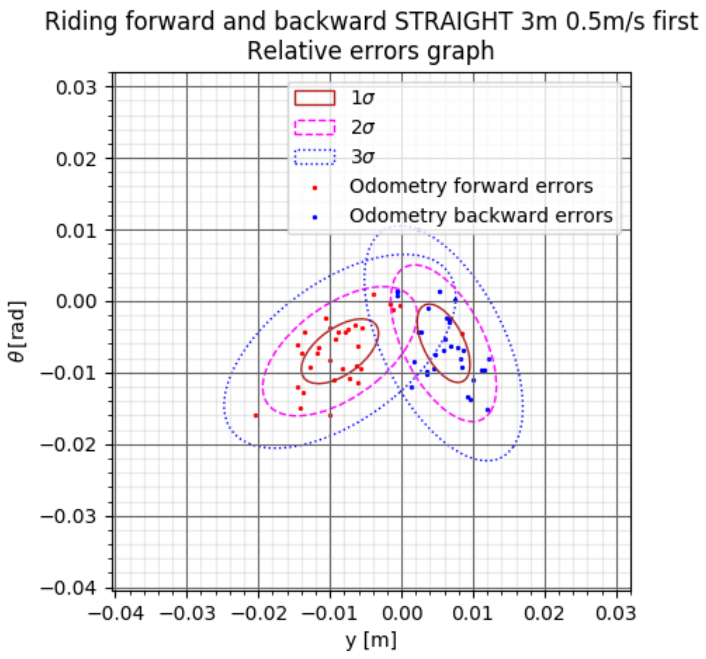


Figure 6.31: Plot of y and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (first interval; straight trajectory - null angular velocity)

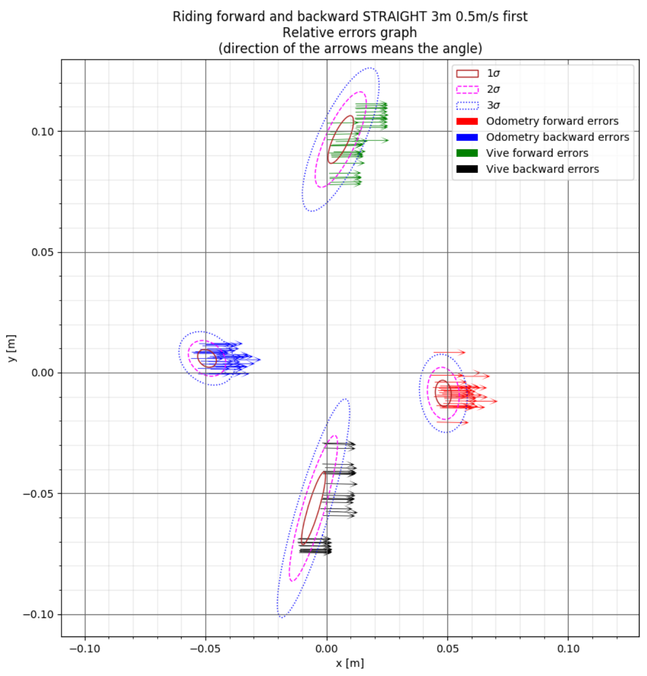


Figure 6.32: Plot of final error vectors of odometries towards the laser tracker reference data and their confidence ellipses (first interval; straight trajectory trajectory - null angular velocity)

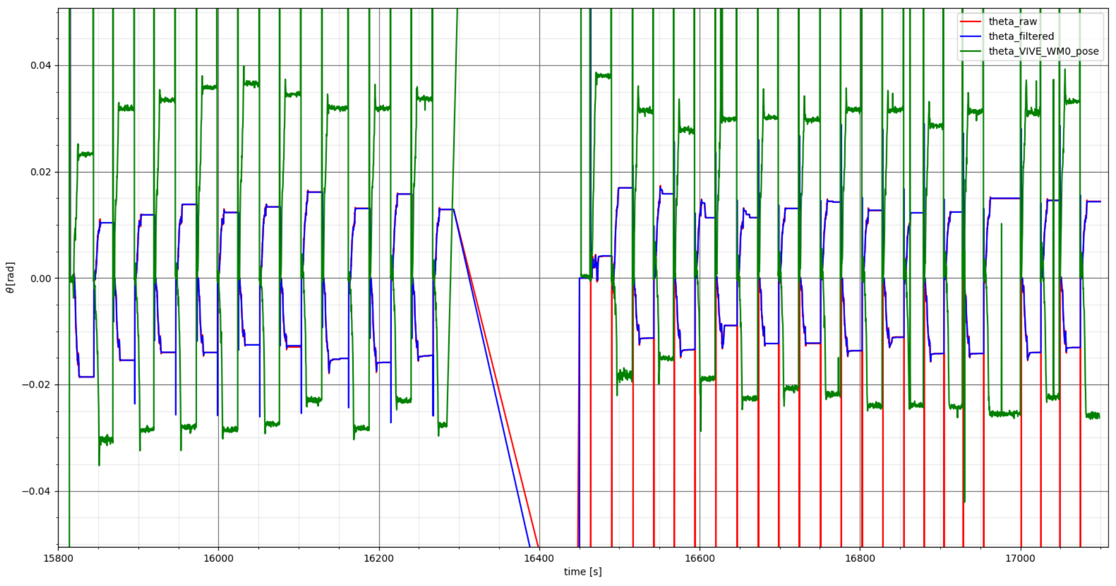


Figure 6.33: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving straight forward and backward (the entire second interval; note that here, the filtered odometry has the same values as the raw encoder odometry)

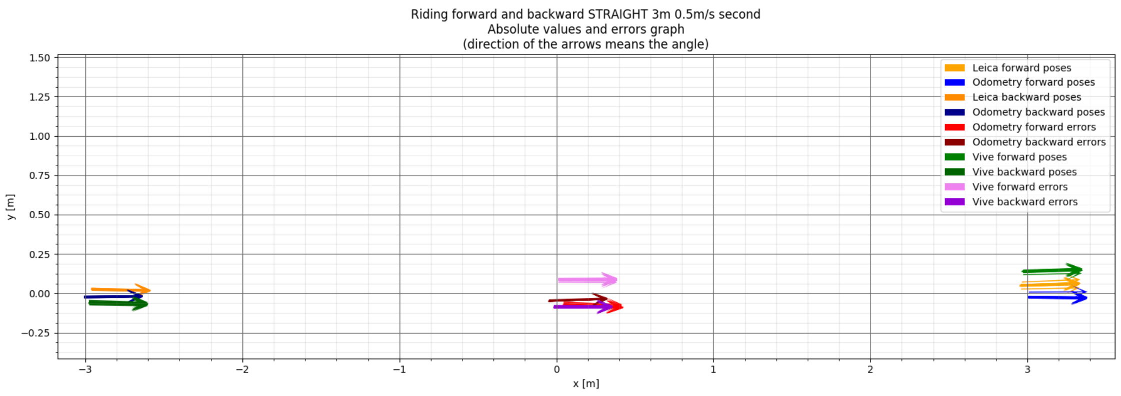


Figure 6.34: Plot of final poses and error vectors (towards the laser tracker final poses) of the robot's adjusted trajectories of odometries (second interval; straight trajectory - null angular velocity)

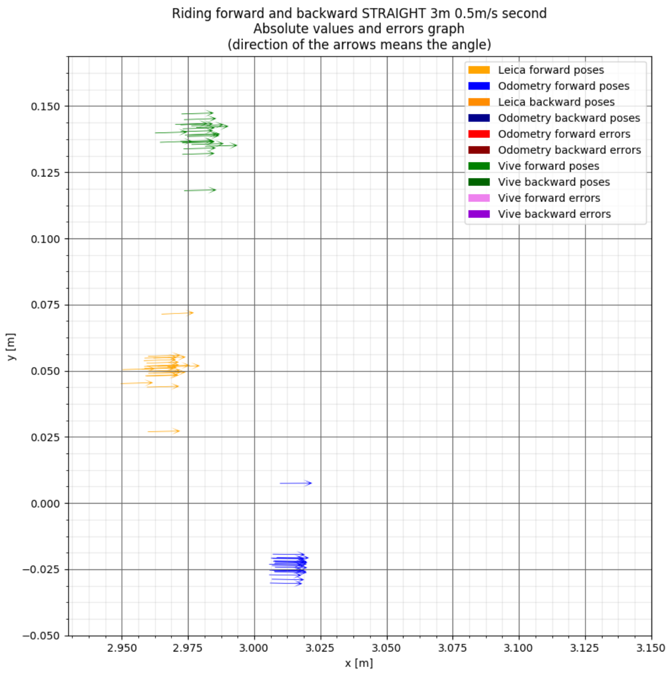


Figure 6.35: Plot of final poses of the robot's adjusted trajectories of odometries (second interval; straight trajectory - null angular velocity, forward detail)

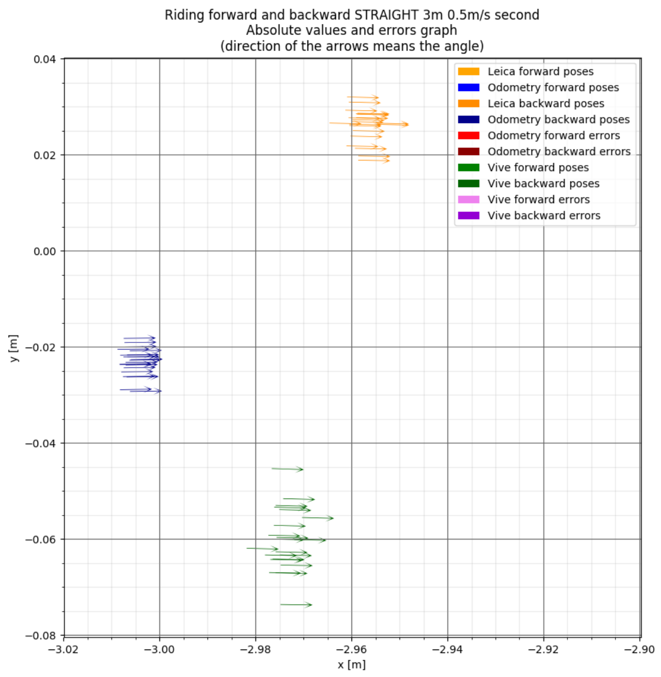


Figure 6.36: Plot of final poses of the robot's adjusted trajectories of odometries (second interval; straight trajectory - null angular velocity, backward detail)

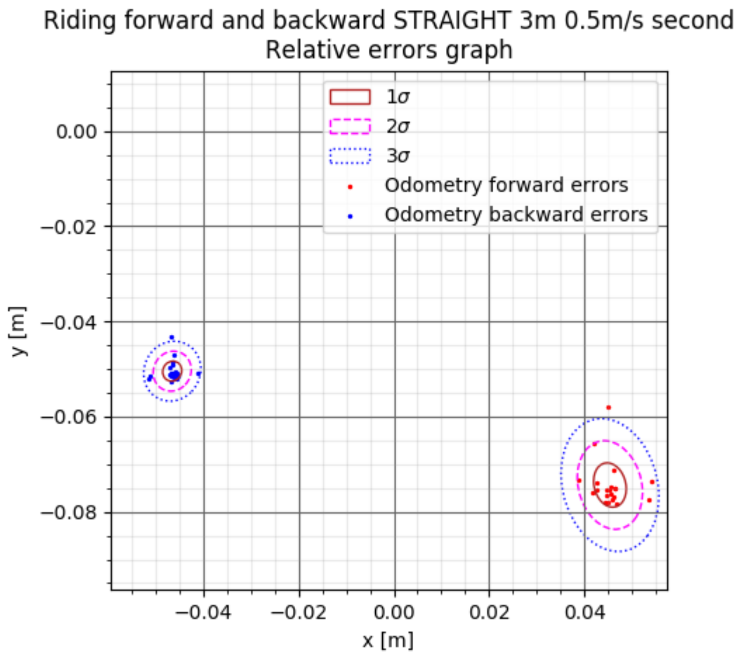


Figure 6.37: Plot of x and y components of relative errors of the final poses of the robot's adjusted trajectories of odometries (second interval; straight trajectory - null angular velocity)

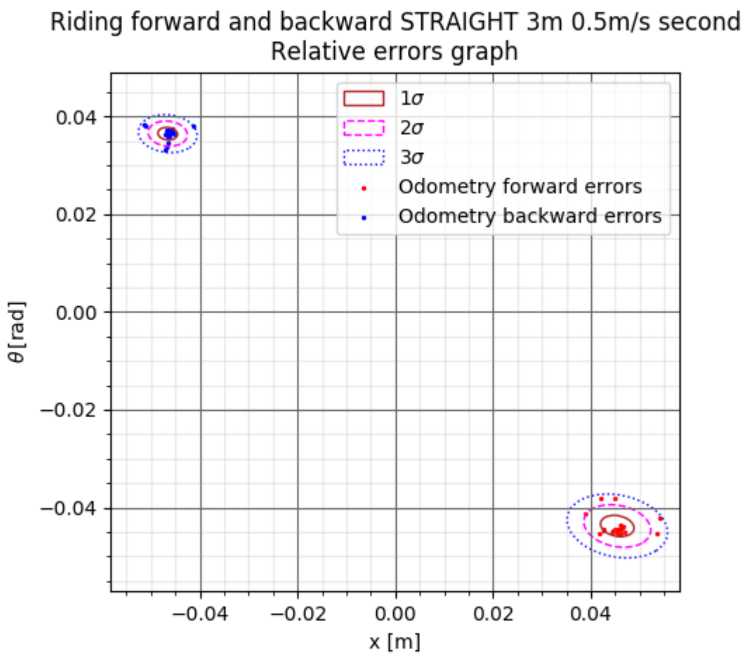


Figure 6.38: Plot of x and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (second interval; straight trajectory - null angular velocity)

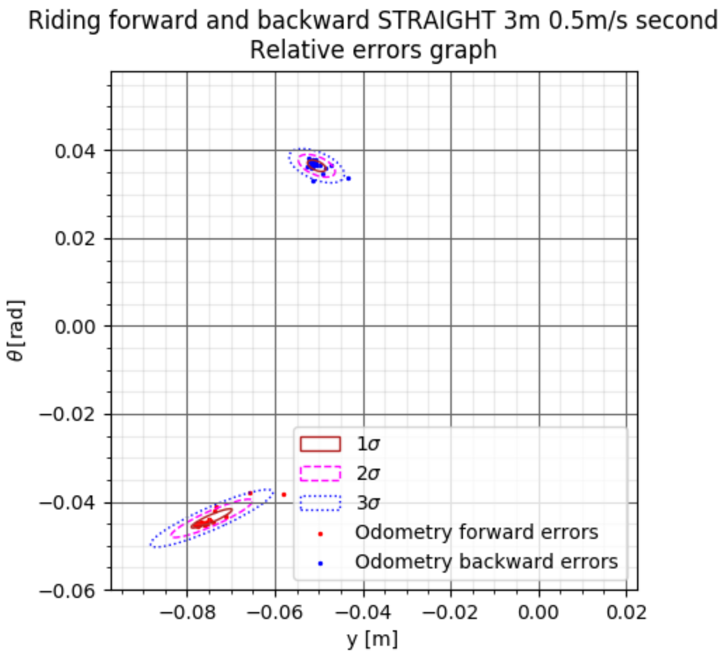


Figure 6.39: Plot of y and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (second interval; straight trajectory - null angular velocity)

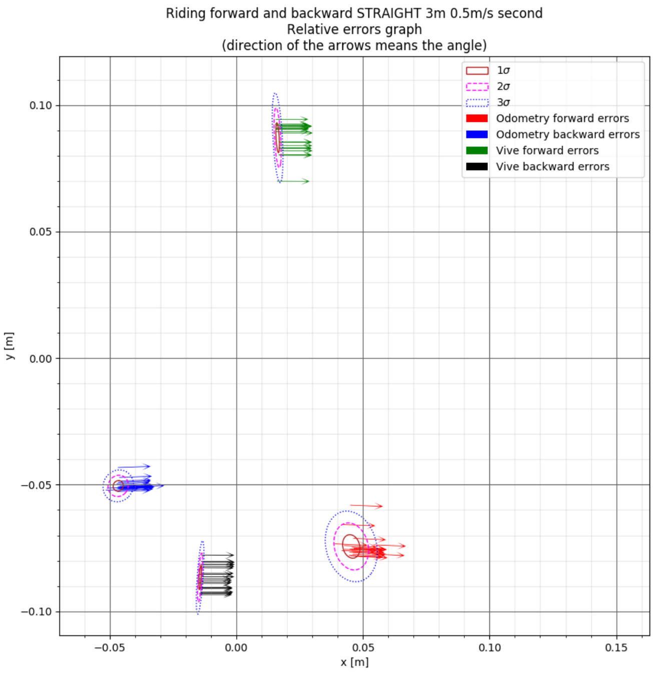


Figure 6.40: Plot of final error vectors of odometries towards the laser tracker reference data and their confidence ellipses (second interval; straight trajectory - null angular velocity)

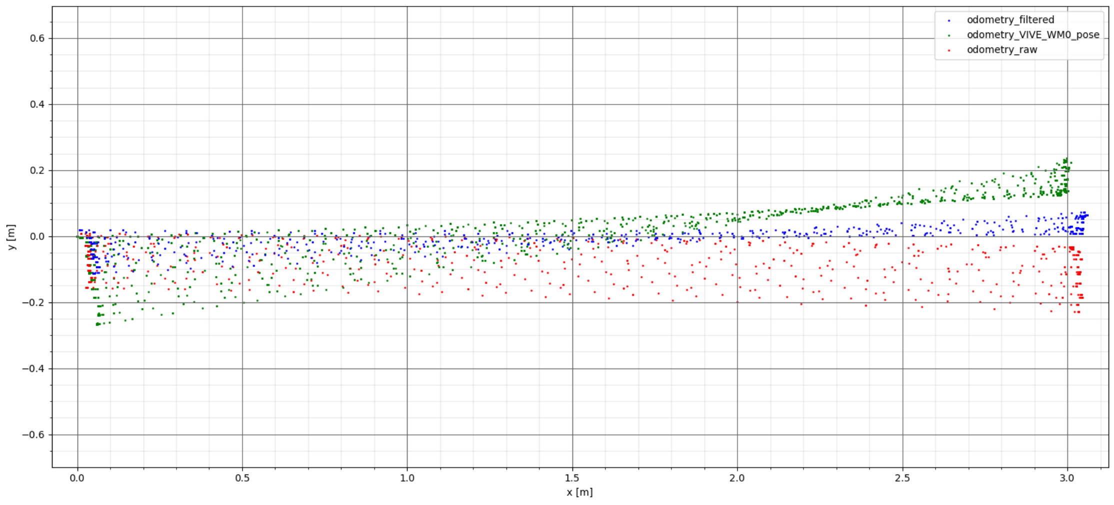


Figure 6.41: Plot of XY-data the robot's trajectories of unadjusted odometries when repeatedly driving straight forward and backward (first 12 successful rounds)

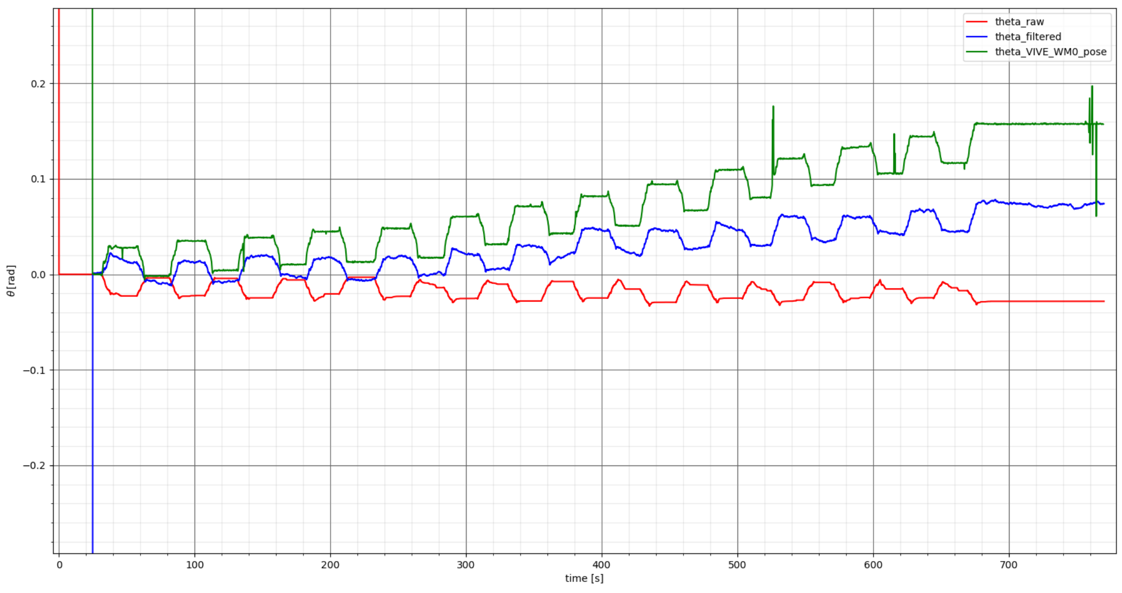


Figure 6.42: Plot of θ-data of the robot's trajectories of unadjusted odometries when repeatedly driving forward and backward (first 12 successful rounds)

Results from the part with properly initialized IMU. All the data from the part of the experiment, where the IMU had been properly initialized and thus had led to smaller error vectors, have been measured and processed. We can see from the plots of relative error vectors in the figures 6.29 through 6.31 that **the distribution of the relative error vector can be approximated by a 3-dimensional normal distribution** (as of x, y, θ). The claiming is based on the fact that all the 3 pairs of the x-components, y-components and θ-components of the error vectors **lie mostly within the 2σ-ellipses of confidence**, of course with some appearances within the 3σ-ellipses, and at the same time it is obvious that **they appear to be evenly distributed and do not form visible distinct sets of points**. There are also some occasional outliers, but they are so few that we can neglect them for the approximation purposes of this work. Eventually the θ-components appear to keep normally distributed around a certain mean value. Therefore the probabilistic distribution of the EKF-filtered odometry of the relative error vector for this set of velocities during the forward movement

- $v_x = 0.5\,\mathrm{m/s}$
- $v_\theta = 0.0\,\mathrm{rad/s}$

with properly initialized IMU can be characterized by the following statistics:

1. **The forward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu}_{\boldsymbol{e}}(0.5, 0, \text{true}) \doteq \begin{bmatrix} 0.0484\,\mathrm{m} & -0.0086\,\mathrm{m} & -0.0070\,\mathrm{rad} \end{bmatrix}^T \tag{6.3}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{e}}(0.5, 0, \text{true}) \doteq \begin{bmatrix} 10.540 \cdot 10^{-6}\,\mathrm{m}^2 & -1.810 \cdot 10^{-6}\,\mathrm{m}^2 & -2.978 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ -1.810 \cdot 10^{-6}\,\mathrm{m}^2 & 28.213 \cdot 10^{-6}\,\mathrm{m}^2 & 13.164 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ -2.978 \cdot 10^{-6}\,\mathrm{rad \cdot m} & 13.164 \cdot 10^{-6}\,\mathrm{rad \cdot m} & 19.737 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.4}$$

2. **The backward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu}_{\boldsymbol{e}}(-0.5, 0, \text{true}) = \begin{bmatrix} -0.0493\,\mathrm{m} & 0.0059\,\mathrm{m} & -0.0059\,\mathrm{rad} \end{bmatrix}^T \tag{6.5}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{e}}(-0.5, 0, \text{true}) = \begin{bmatrix} 14.533 \cdot 10^{-6}\,\mathrm{m}^2 & -3.483 \cdot 10^{-6}\,\mathrm{m}^2 & 10.948 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ -3.483 \cdot 10^{-6}\,\mathrm{m}^2 & 13.197 \cdot 10^{-6}\,\mathrm{m}^2 & -10.703 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ 10.948 \cdot 10^{-6}\,\mathrm{rad \cdot m} & -10.703 \cdot 10^{-6}\,\mathrm{rad \cdot m} & 29.023 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.6}$$

Results from the part with improperly initialized IMU. All the data from the part of the experiment, where the IMU had not been properly initialized and thus had led to larger error vectors, have been measured and processed. We can see from the plots of relative error vectors in the figures 6.37 through 6.39 that **the relative error vector can be approximated**

by a 3-dimensional normal distribution in the same way and from the same reason as the error vectors from the interval of properly initialized IMU. Therefore the probabilistic distribution of the EKF-filtered odometry of the relative error vector for this set of velocities during the forward movement

- $v_x = 0.5\,\mathrm{m/s}$
- $v_\theta = 0.0\,\mathrm{rad/s}$

with improperly initialized IMU can be characterized by the following statistics:

1. **The forward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu_e}(0.5, 0, \mathrm{true}) = \begin{bmatrix} 0.0454\,\mathrm{m} & -0.0744\,\mathrm{m} & -0.0438\,\mathrm{rad} \end{bmatrix}^T \tag{6.7}$$

$$\boldsymbol{\Sigma_e}(0.5, 0, \mathrm{true}) = \begin{bmatrix} 11.176 \cdot 10^{-6}\,\mathrm{m}^2 & -2.600 \cdot 10^{-6}\,\mathrm{m}^2 & -1.395 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ -2.600 \cdot 10^{-6}\,\mathrm{m}^2 & 20.601 \cdot 10^{-6}\,\mathrm{m}^2 & 8.722 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ -1.395 \cdot 10^{-6}\,\mathrm{rad \cdot m} & 8.722 \cdot 10^{-6}\,\mathrm{rad \cdot m} & 4.522 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.8}$$

2. **The backward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu_e}(-0.5, 0, \mathrm{true}) = \begin{bmatrix} -0.0466\,\mathrm{m} & -0.0505\,\mathrm{m} & 0.0365\,\mathrm{rad} \end{bmatrix}^T \tag{6.9}$$

$$\boldsymbol{\Sigma_e}(-0.5, 0, \mathrm{true}) = \begin{bmatrix} 3.814 \cdot 10^{-6}\,\mathrm{m}^2 & 0.299 \cdot 10^{-6}\,\mathrm{m}^2 & -0.237 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ 0.299 \cdot 10^{-6}\,\mathrm{m}^2 & 4.204 \cdot 10^{-6}\,\mathrm{m}^2 & -1.341 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ -0.237 \cdot 10^{-6}\,\mathrm{rad \cdot m} & -1.341 \cdot 10^{-6}\,\mathrm{rad \cdot m} & 1.590 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.10}$$

For the purposes and scope of this work, the statistics of the tracking system odometry will not be studied, as the graphs depicting them for the straight direction have provided us with a sufficient overview of those statistics and the tracking system will not be present during the robot's ordinary operation. Therefore the tracking system **characteristics will not be plotted anymore in the graphs corresponding to the following trajectories**, besides other reasons in order to make the graphs also more legible and clearer.

Driving 3 meters forward and backward at 0.5 m/s with angular velocity of 0.1 rad/s

This trajectory was driven in **57 successfully recorded loops**, with only **1 rejected outlier** in the forward direction. Particularly this experiment was performed with 2 several minutes long breaks after first 20 and first 38 successfully measured entire loops, respectively, during which the robot was not rebooted, unlike during the the significantly longer breaks during recording data for the straight trajectory described above, which thus did not lead to the improper initialization of the IMU, described above. Therefore **only data of the odometries with properly initialized IMU were recorded for this trajectory**, which is apparent from the plots of θ-components of the odometries in the figures 6.44 and 6.45. The reason for this claiming is that the values of the θ-component of the filtered odometry in the plots appear to be significantly different from the values of the raw encoder odometry, which has been marked to be an evidence that the IMU is properly initialized.

The **plot of the robot's paths** (at least their final poses) read from the **raw encoder odometry** (only in some of the figures) and the **filtered odometry**, **adjusted always to the null position when changing its direction from** forward to backward or vice versa (assumed just after having had its pose measured by the laser tracker), with the **respective graphs of errors**, can be seen in the figures 6.43[2] through 6.52. Also final poses estimated from the laser tracker measurement appear in some of the graphs (marked in the legends of the particular graphs).

The graphs show that **the robot keeps again subtly steering slightly more left it the direction of its driving**, in addition to the commanded angular velocity. In reality it results in a **sawtooth-like trajectory**, which may be seen in the plot of XY-data of odometries in the figure 6.43 from the fact that the forward branch is more tilted in the positive y-direction than the backward branch. This forces the user to stop the robot, lead it back to the original pose and start the measurement process again after a certain count of loops, because otherwise the robot would get to the angles exceeding those under which the laser tracker is able to measure the positions of the attached retroreflectors. It is the cause for the mentioned several minutes long breaks.

[2]Please note that the outlier points in the graphs of trajectories are caused only by an imperfection in the data processing script.

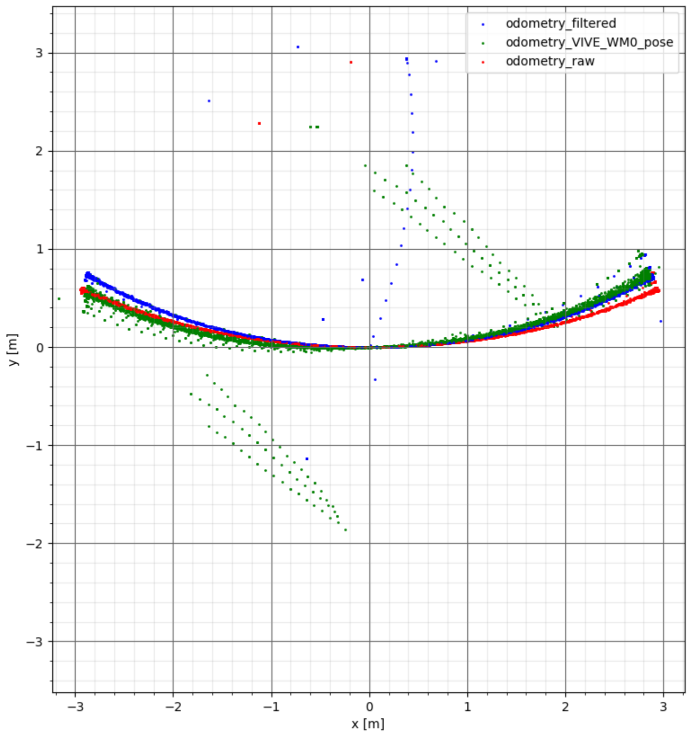


Figure 6.43: Plot of XY-data of the robot's adjusted trajectories when repeatedly driving forward and backward (forward angular velocity of 0.1 rad/s)

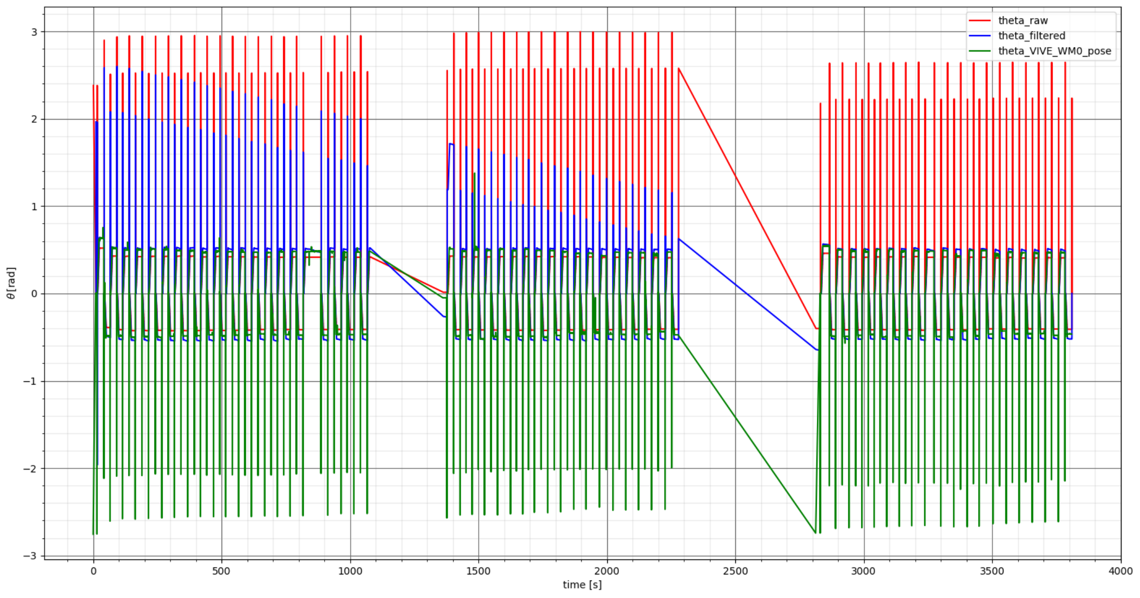


Figure 6.44: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving forward and backward (forward angular velocity of 0.1 rad/s)

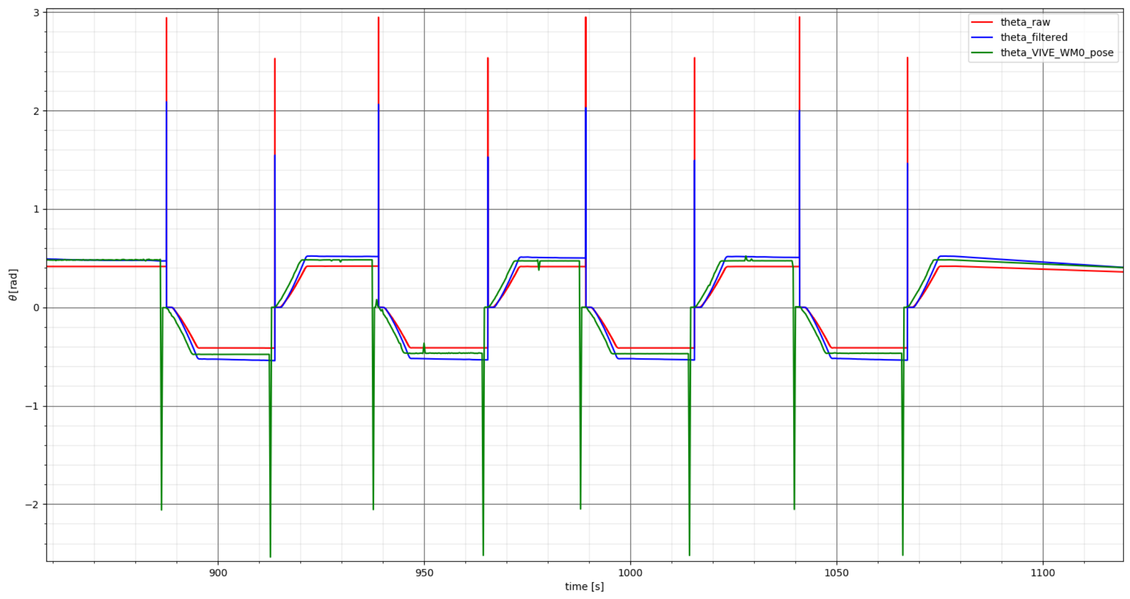


Figure 6.45: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving forward and backward (detail of a sub-interval between times of 850 s and 1100 s; forward angular velocity of 0.1 rad/s)

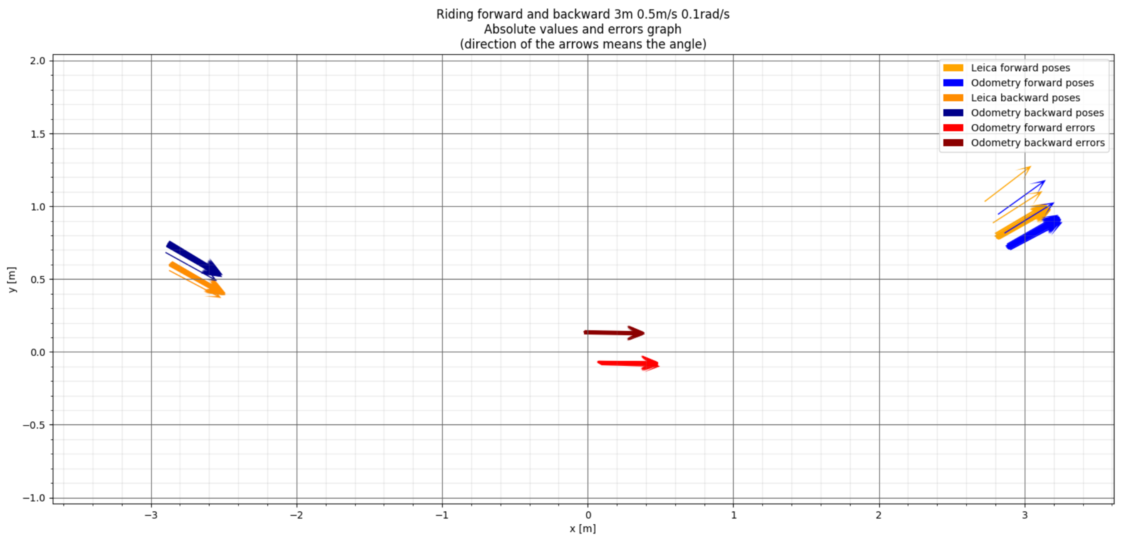


Figure 6.46: Plot of final poses and error vectors (towards the laser tracker final poses) of the robot's adjusted trajectories of the filtered odometry (both intervals; forward angular velocity of 0.1 rad/s)

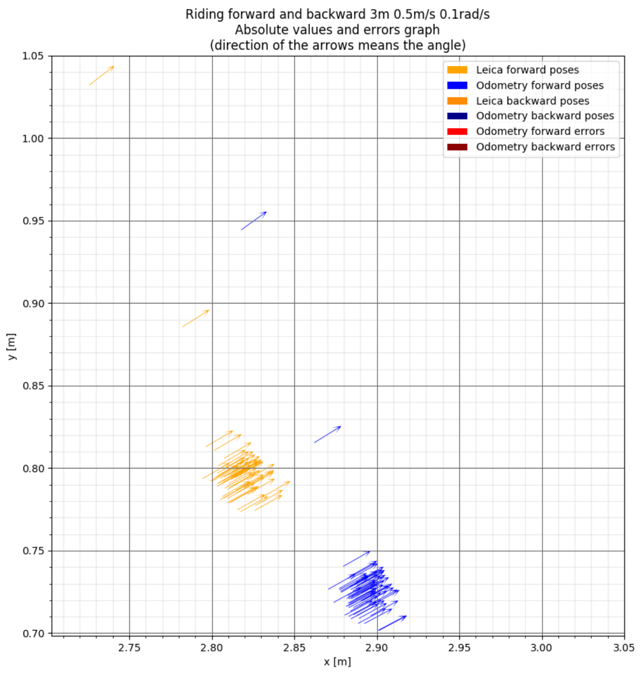


Figure 6.47: Plot of final poses of the robot's adjusted trajectories of the filtered odometry (forward angular velocity of 0.1 rad/s, forward detail)

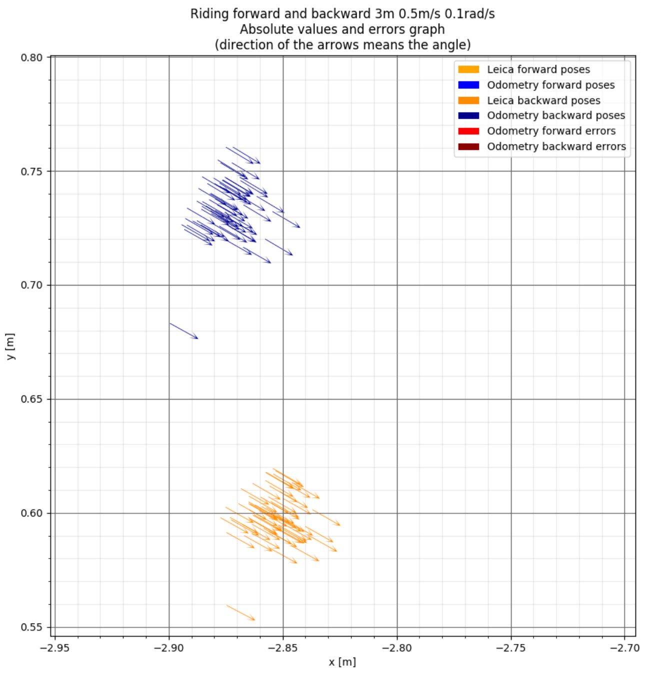


Figure 6.48: Plot of final poses of the robot's adjusted trajectories of the filtered odometry (forward angular velocity of 0.1 rad/s, backward detail)

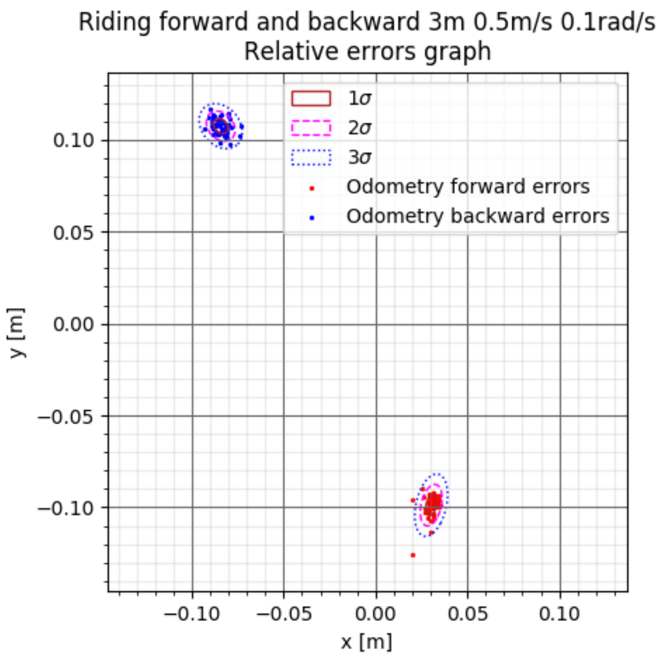


Figure 6.49: Plot of x and y components of relative errors of the final poses of the robot's adjusted trajectories of odometries (forward angular velocity of 0.1 rad/s)

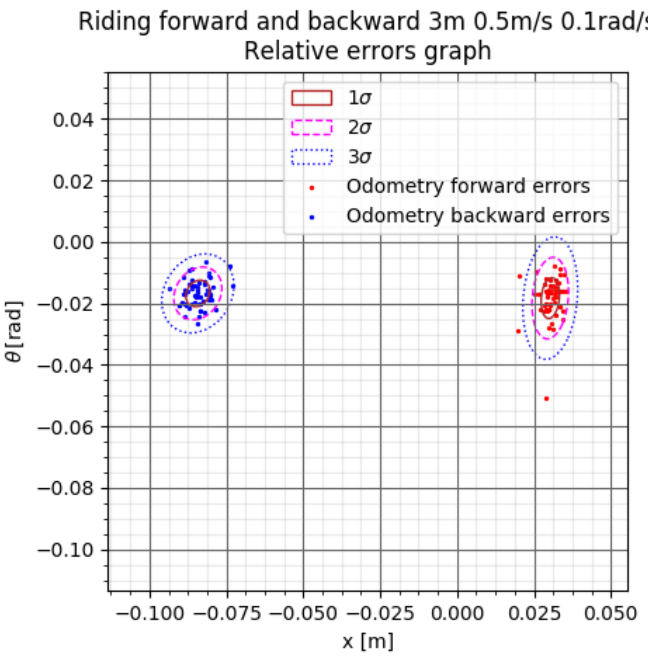


Figure 6.50: Plot of x and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (forward angular velocity of 0.1 rad/s)

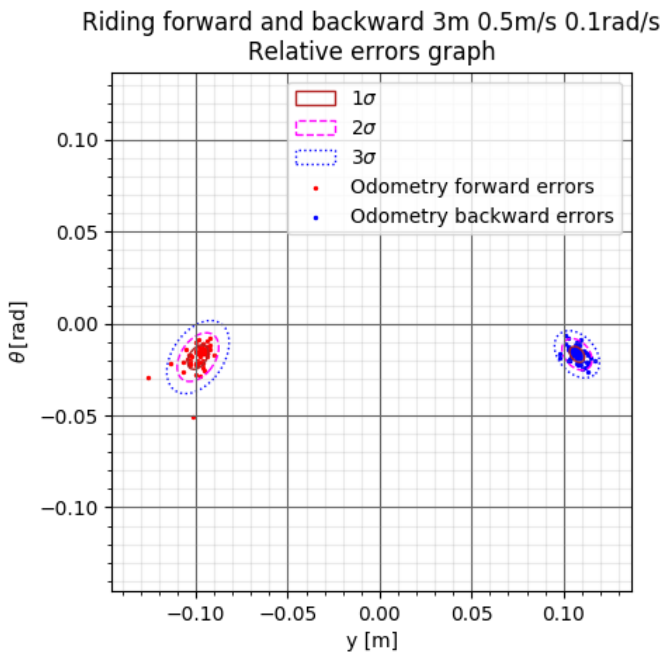


Figure 6.51: Plot of y and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (forward angular velocity of 0.1 rad/s)

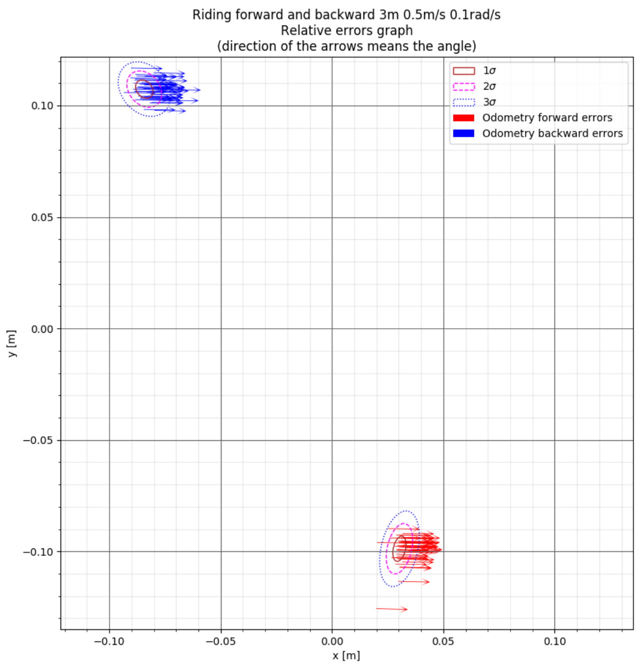


Figure 6.52: Plot of final error vectors of odometries towards the laser tracker reference data and their confidence ellipses (forward angular velocity of 0.1 rad/s)

Results of the experiment for this set of velocities. All the data from the experiment have been measured and processed. We can see from the plots of relative error vectors in the figures 6.49 through 6.51 that **the relative error vector can be approximated by a 3-dimensional normal distribution** in the same way and from the same reason described at the end of the previous subsection on the page 96. Therefore the probabilistic distribution of the EKF-filtered odometry of the relative error vector for this set of velocities during the forward movement

- $v_x = 0.5\,\mathrm{m/s}$
- $v_\theta = 0.1\,\mathrm{rad/s}$

with properly initialized IMU can be characterized by the following statistics:

1. **The forward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu_e}(0.5, 0.1, \mathrm{true}) = \begin{bmatrix} 0.0303\,\mathrm{m} & -0.0988\,\mathrm{m} & -0.0183\,\mathrm{rad} \end{bmatrix}^T \tag{6.11}$$

$$\boldsymbol{\Sigma_e}(0.5, 0.1, \mathrm{true}) = \begin{bmatrix} 8.675 \cdot 10^{-6}\,\mathrm{m}^2 & 5.523 \cdot 10^{-6}\,\mathrm{m}^2 & 3.078 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} \\ 5.523 \cdot 10^{-6}\,\mathrm{m}^2 & 31.378 \cdot 10^{-6}\,\mathrm{m}^2 & 14.943 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} \\ 3.078 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} & 14.943 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} & 43.138 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.12}$$

2. **The backward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu_e}(-0.5, -0.1, \mathrm{true}) = \begin{bmatrix} -0.0843\,\mathrm{m} & 0.1073\,\mathrm{m} & -0.0168\,\mathrm{rad} \end{bmatrix}^T \tag{6.13}$$

$$\boldsymbol{\Sigma_e}(-0.5, -0.1, \mathrm{true}) = \begin{bmatrix} 15.025 \cdot 10^{-6}\,\mathrm{m}^2 & -4.117 \cdot 10^{-6}\,\mathrm{m}^2 & 3.251 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} \\ -4.117 \cdot 10^{-6}\,\mathrm{m}^2 & 16.345 \cdot 10^{-6}\,\mathrm{m}^2 & -6.048 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} \\ 3.251 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} & -6.048 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} & 18.006 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.14}$$

These results correspond to the data after having removed **1 outlier measured in the forward direction**.

Driving 3 meters forward and backward at 0.5 m/s with angular velocity of -0.1 rad/s

This trajectory was driven in **73 successfully recorded loops**. Particularly this experiment was performed with 3 several minutes long breaks after first 8, first 17 and first 43 successfully measured entire loops, respectively, during which the robot was not rebooted, unlike during the the significantly longer breaks during recording data for the straight trajectory described above, which thus did not lead to the improper initialization of the IMU, described above. Therefore **only data of the odometries with <u>improperly</u> initialized IMU were recorded for this trajectory**, which is apparent from the plots of θ-components of the odometries in the figures 6.54 through 6.64. The reason for this claiming is that the values of the θ-component of the filtered odometry and the values of the θ-component of the raw encoder odometry in the plots appear to be identical, which has been marked to be an evidence that the IMU is properly initialized.

The **plot of the robot's paths** (at least their final poses) read from the **raw encoder odometry** (only in some of the figures) and the **filtered odometry**, **adjusted always to the null position when changing its direction from** forward to backward or vice versa, with the **respective graphs of errors**, can be seen in the figures 6.53[3] through 6.57. Also final poses estimated from the laser tracker measurement appear in graphs in the figures 6.58 through 6.60 (marked in the legends of the graphs).

The graphs show that **the robot keeps again subtly steering slightly more left it the direction of its driving**, in addition to the commanded angular velocity, which has the same consequences as during the previous parts of the experiment.

[3]Please note that the outlier points in the graphs of trajectories are caused only by an imperfection in the data processing script.

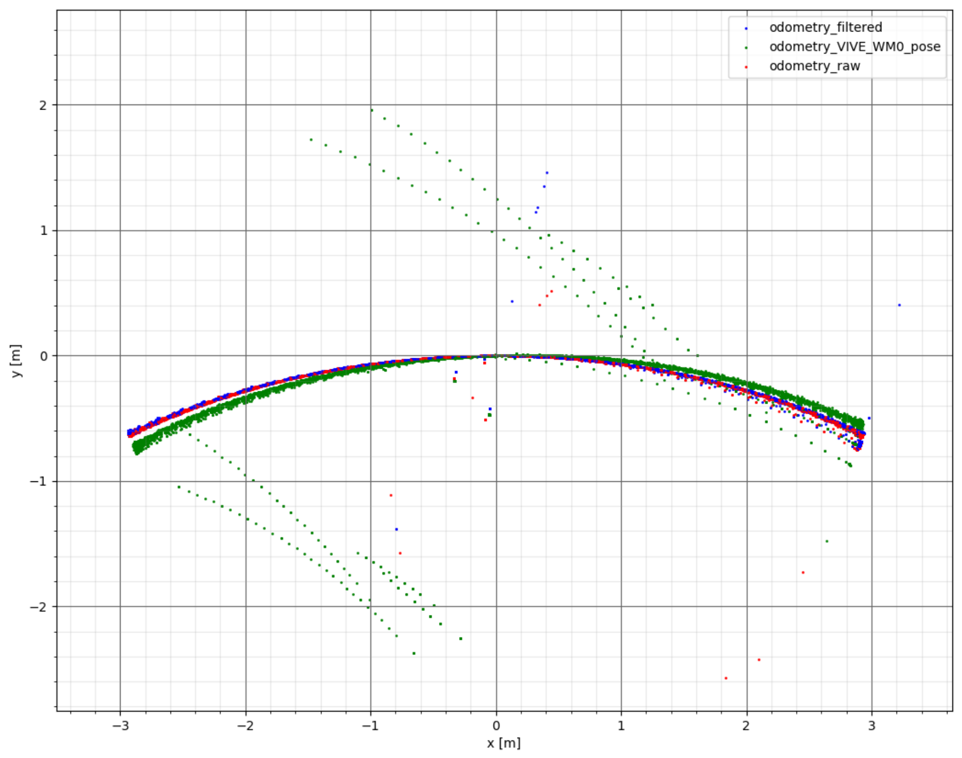


Figure 6.53: Plot of XY-data of the robot's adjusted trajectories when repeatedly driving forward and backward (forward angular velocity of -0.1 rad/s)

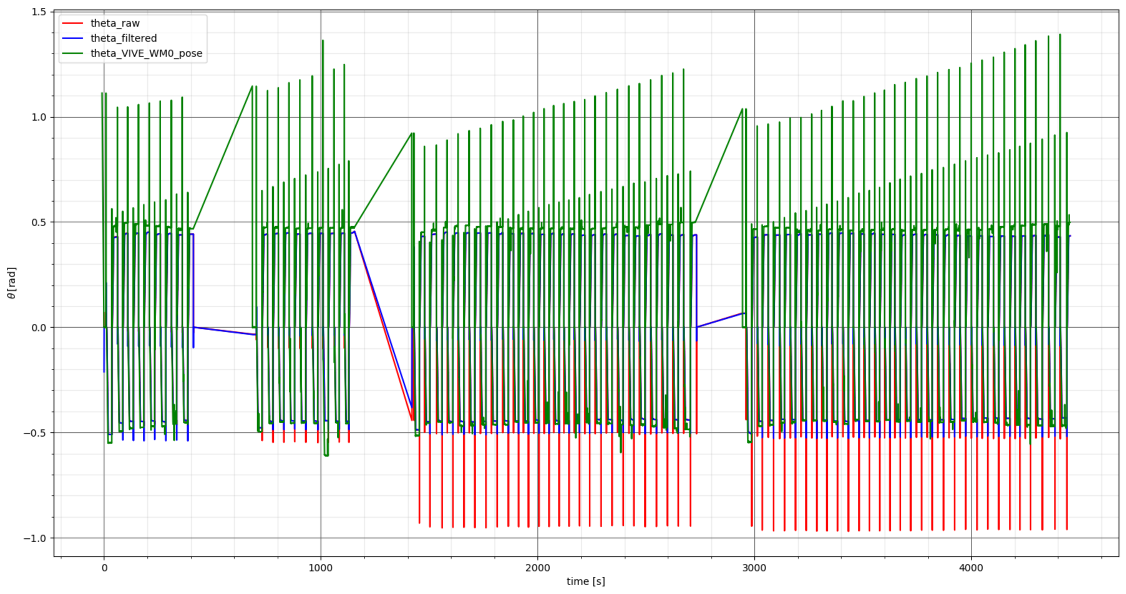


Figure 6.54: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving forward and backward (forward angular velocity of -0.1 rad/s)

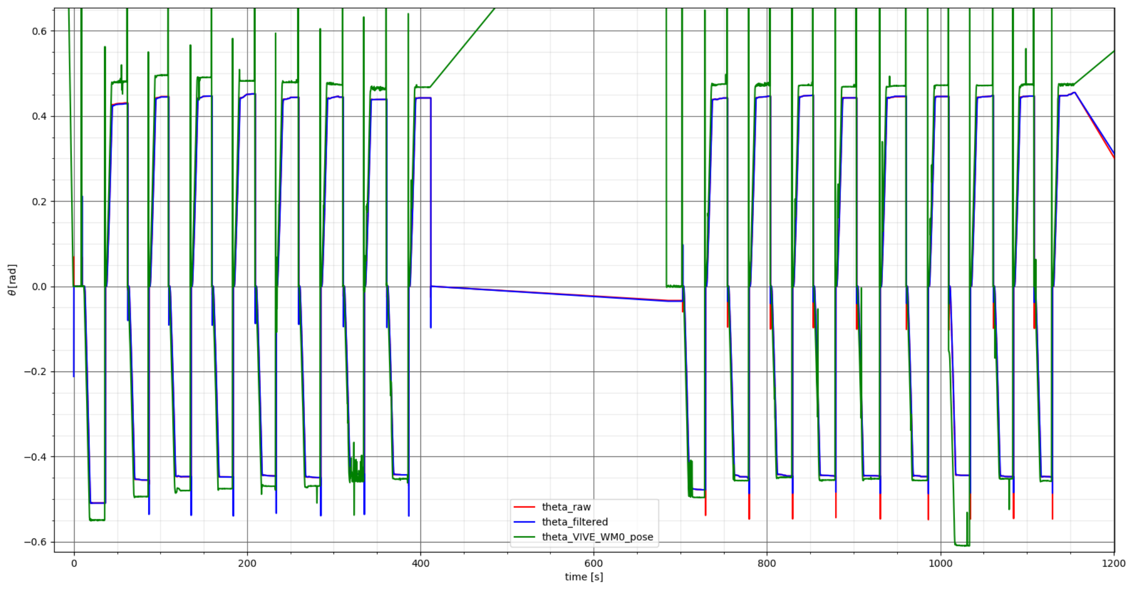


Figure 6.55: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving forward and backward (detail of first 2 sub-intervals between times of 0 s and 1200 s; forward angular velocity of -0.1 rad/s)

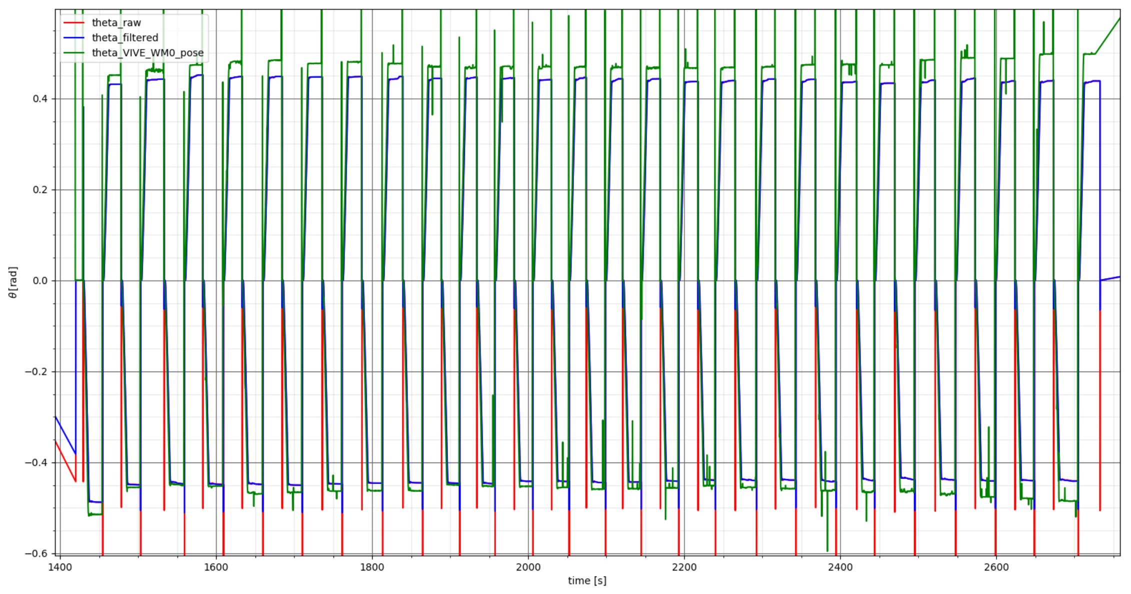


Figure 6.56: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving forward and backward (detail of the 3rd sub-interval between times of 1400 s and 2800 s; forward angular velocity of -0.1 rad/s)

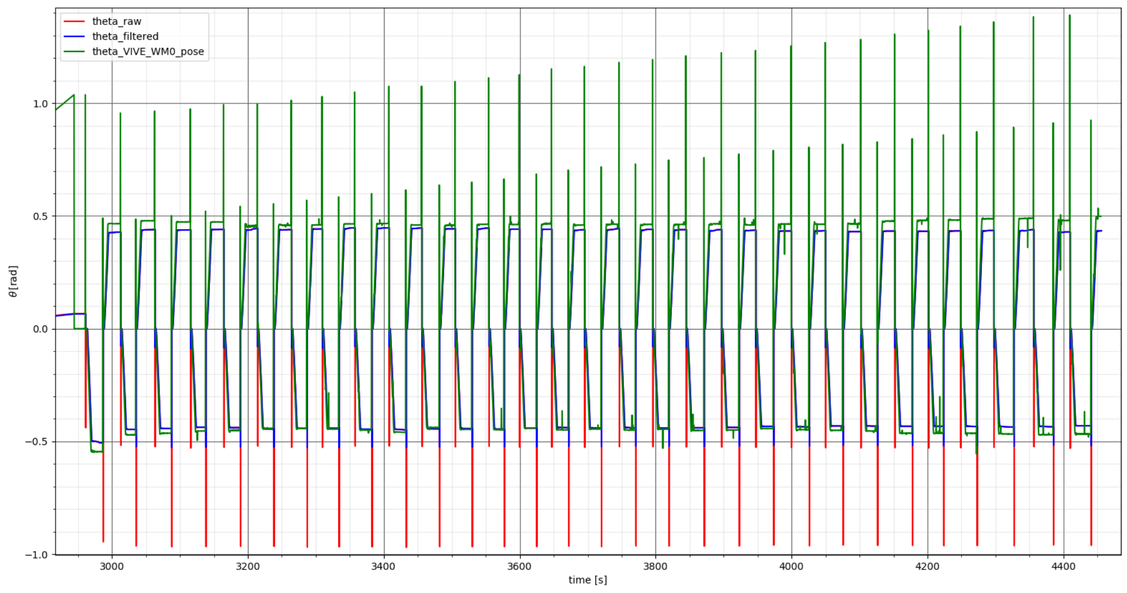


Figure 6.57: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving forward and backward (detail of the 4th sub-interval between times of 2900 s and 4500 s; forward angular velocity of -0.1 rad/s)

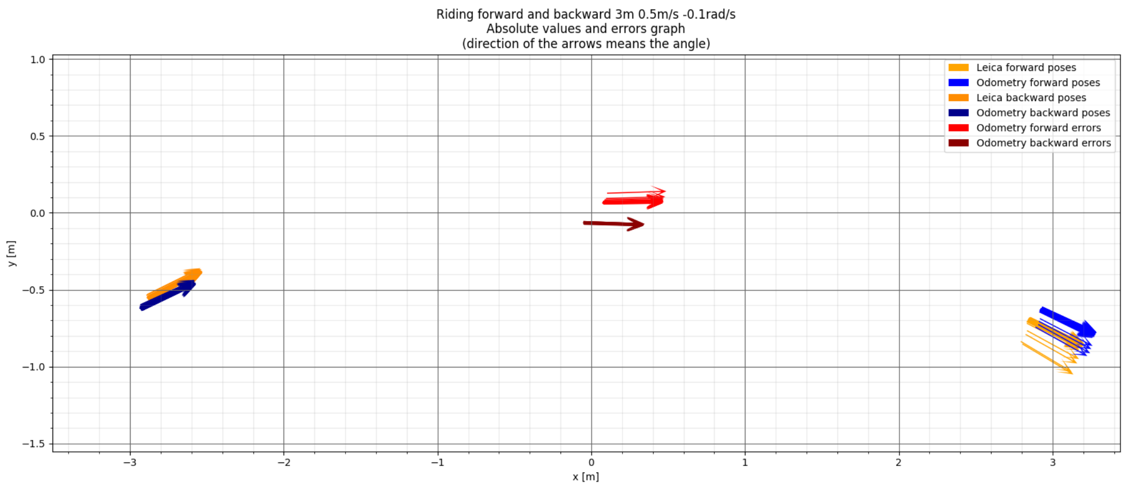


Figure 6.58: Plot of final poses and error vectors (towards the laser tracker final poses) of the robot's adjusted trajectories of the filtered odometry (forward angular velocity of -0.1 rad/s)

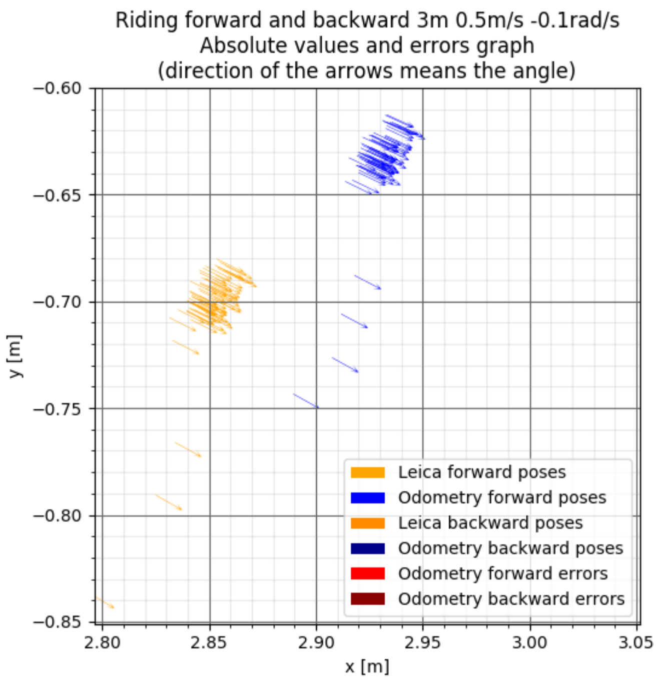


Figure 6.59: Plot of final poses of the robot's adjusted trajectories of the filtered odometry (forward angular velocity of −0.1 rad/s, forward detail)

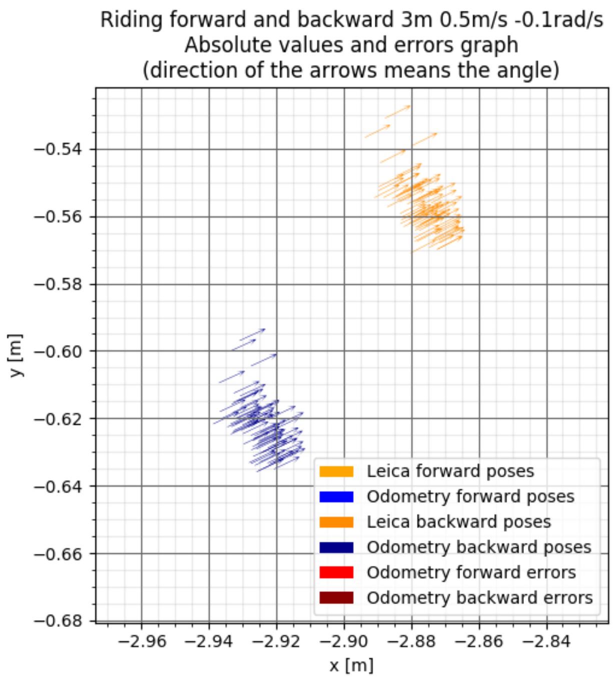


Figure 6.60: Plot of final poses of the robot's adjusted trajectories of the filtered odometry (forward angular velocity of −0.1 rad/s, backward detail)

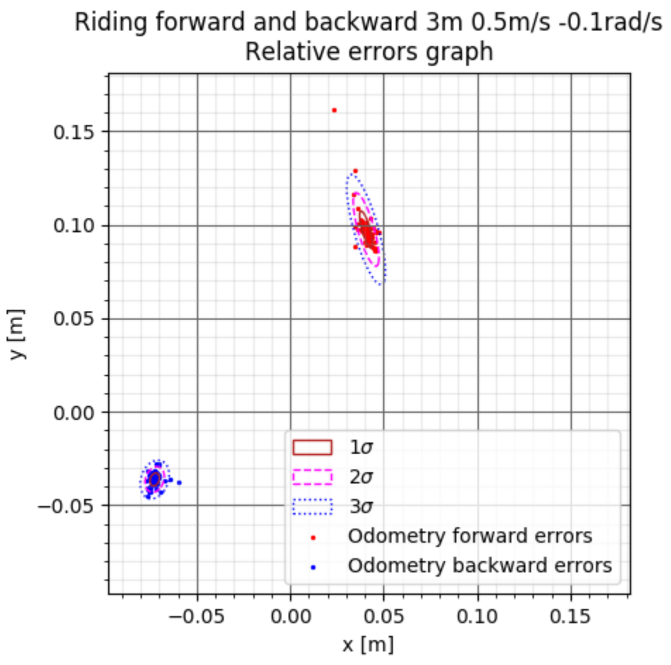


Figure 6.61: Plot of x and y components of relative errors of the final poses of the robot's adjusted trajectories of odometries (forward angular velocity of -0.1 rad/s)

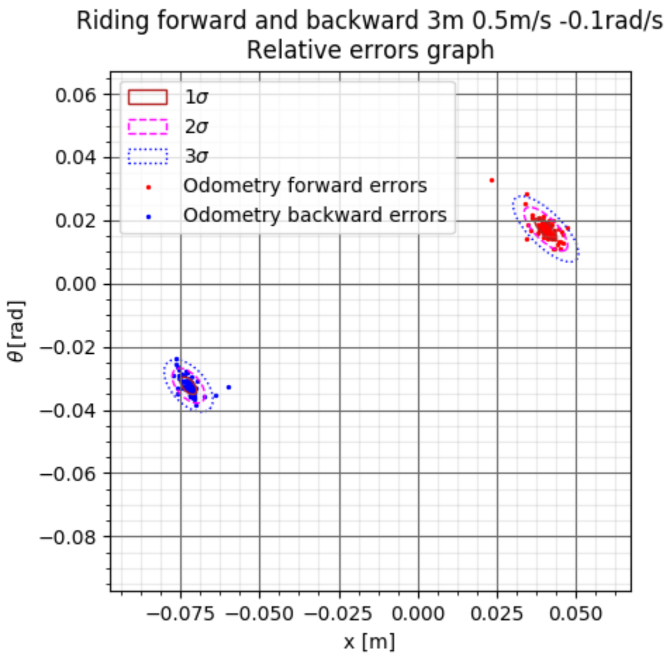


Figure 6.62: Plot of x and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (forward angular velocity of -0.1 rad/s)

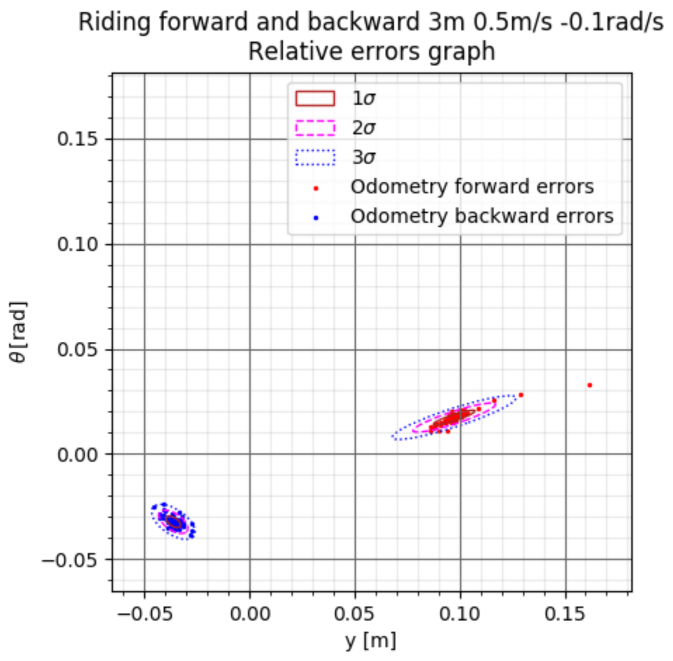


Figure 6.63: Plot of y and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (forward angular velocity of -0.1 rad/s)

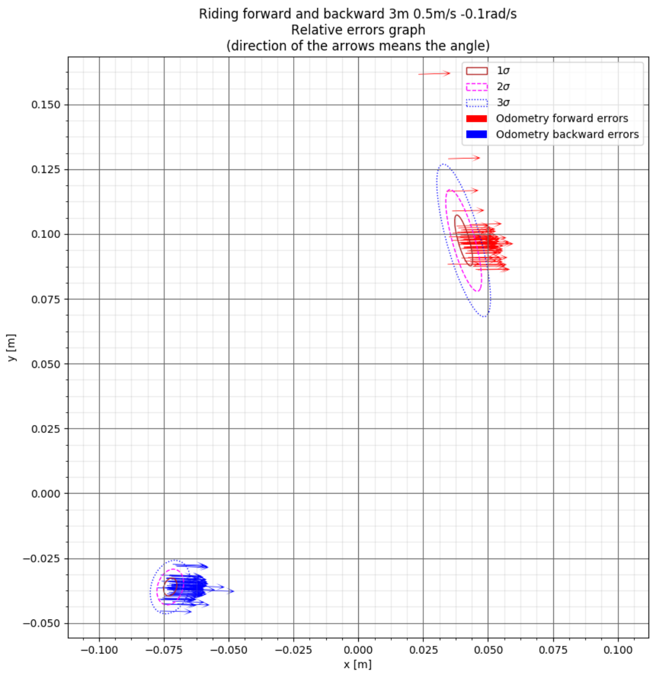


Figure 6.64: Plot of final error vectors of odometries towards the laser tracker reference data and their confidence ellipses (forward angular velocity of -0.1 rad/s)

Results of the experiment for this set of velocities. All the data from the experiment have been measured and processed. We can see from the plots of relative error vectors in the figures 6.61 through 6.63 that **the relative error vector can be approximated by a 3-dimensional normal distribution** in the same way and from the same reason described at the end of the subsection about the part of this experiment driven with null angular velocity on the page 96. Therefore the probabilistic distribution of the EKF-filtered odometry of the relative error vector for this set of velocities during the forward movement

- $v_x = 0.5\,\mathrm{m/s}$
- $v_\theta = -0.1\,\mathrm{rad/s}$

with improperly initialized IMU can be characterized by the following statistics:

1. **The forward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu_e}(0.5, -0.1, \mathrm{false}) = \begin{bmatrix} 0.0406\,\mathrm{m} & 0.0975\,\mathrm{m} & 0.0172\,\mathrm{rad} \end{bmatrix}^T \tag{6.15}$$

$$\boldsymbol{\Sigma_e}(0.5, -0.1, \mathrm{false}) = \begin{bmatrix} 11.768 \cdot 10^{-6}\,\mathrm{m}^2 & -25.337 \cdot 10^{-6}\,\mathrm{m}^2 & -8.731 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} \\ -25.337 \cdot 10^{-6}\,\mathrm{m}^2 & 94.871 \cdot 10^{-6}\,\mathrm{m}^2 & 30.003 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} \\ -8.731 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} & 30.003 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} & 11.835 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.16}$$

2. **The backward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu_e}(-0.5, 0.1, \mathrm{false}) = \begin{bmatrix} -0.0725\,\mathrm{m} & -0.0362\,\mathrm{m} & -0.0324\,\mathrm{rad} \end{bmatrix}^T \tag{6.17}$$

$$\boldsymbol{\Sigma_e}(-0.5, 0.1, \mathrm{false}) = \begin{bmatrix} 6.539 \cdot 10^{-6}\,\mathrm{m}^2 & 2.018 \cdot 10^{-6}\,\mathrm{m}^2 & -3.702 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} \\ 2.018 \cdot 10^{-6}\,\mathrm{m}^2 & 11.665 \cdot 10^{-6}\,\mathrm{m}^2 & -5.332 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} \\ -3.702 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} & -5.332 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} & 7.505 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.18}$$

Unfortunately, the entire experiment was led with **improperly initialized IMU**. It means that the statistics essential to estimate the final dependent probabilistic model, which is primarily meant to be determined for the properly initialized IMU, are not available and will have to be at least approximated, which will be solved later in this work.

Driving 3 meters forward and backward at 0.5 m/s with angular velocity of 0.2 rad/s

This trajectory was driven in **46 successfully recorded loops**. Particularly this experiment was performed with 4 several minutes long breaks after first 6, first 22, first 36 and first 43 successfully measured entire loops, respectively, during which the robot was not rebooted, unlike during the the significantly longer breaks during recording data for the straight trajectory described above, which thus did not lead to the improper initialization of the IMU, described above. Therefore **only data of the odometries with properly initialized IMU were recorded for this trajectory**, which is apparent from the plots of θ-components of the odometries in the figures 6.66 through 6.69. The reason for this claiming is that the values of the θ-component of the filtered odometry in the plots appear to be significantly different from the values of the raw encoder odometry, which has been marked to be an evidence that the IMU is properly initialized.

The **plot of the robot's paths** (at least their final poses) read from the **raw encoder odometry** (only in some of the figures) and the **filtered odometry**, **adjusted always to the null position when changing its direction from** forward to backward or vice versa, with the **respective graphs of errors**, can be seen in the figures 6.65[4] through 6.76. Also final poses estimated from the laser tracker measurement appear in graphs in the figures 6.70 through 6.72 (marked in the legends of the graphs).

The graphs show that **the robot keeps again subtly steering slightly more left it the direction of its driving**, in addition to the commanded angular velocity, which has the same consequences as during the previous parts of the experiment.

[4]Please note that the outlier points in the graphs of trajectories are caused only by an imperfection in the data processing script.

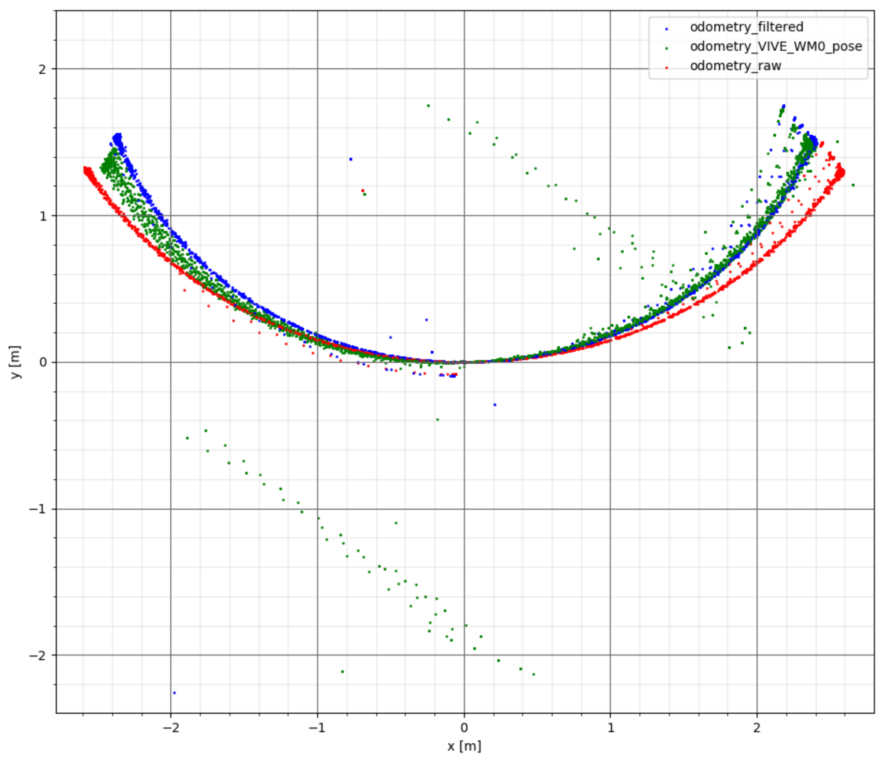


Figure 6.65: Plot of XY-data of the robot's adjusted trajectories when repeatedly driving forward and backward (forward angular velocity of 0.2 rad/s)

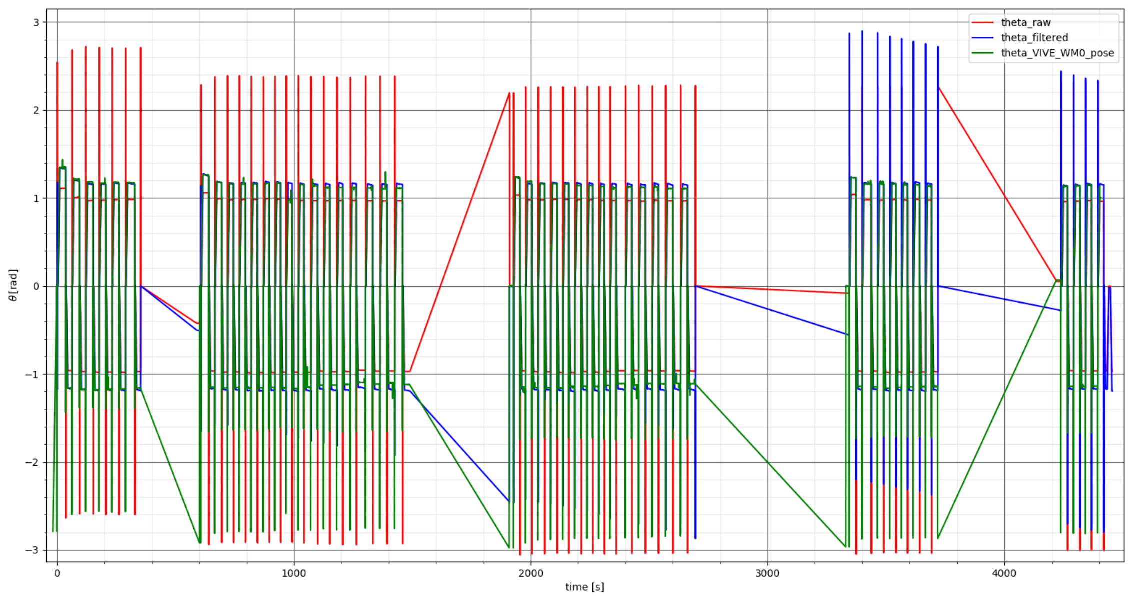


Figure 6.66: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving forward and backward (forward angular velocity of 0.2 rad/s)

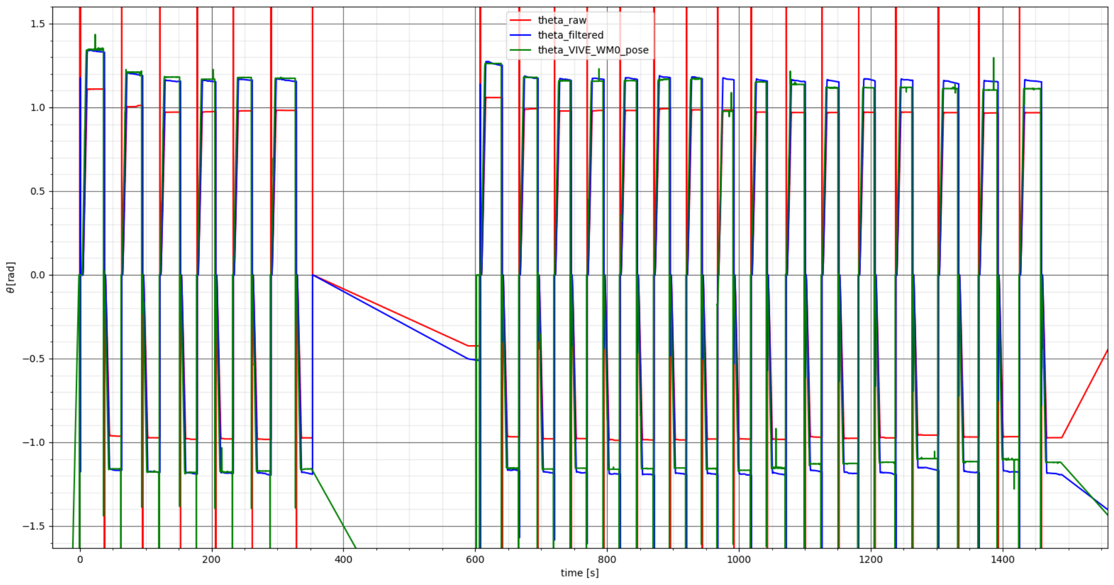


Figure 6.67: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving forward and backward (detail of first 2 sub-intervals between times of 0 s and 1500 s; forward angular velocity of 0.2 rad/s)

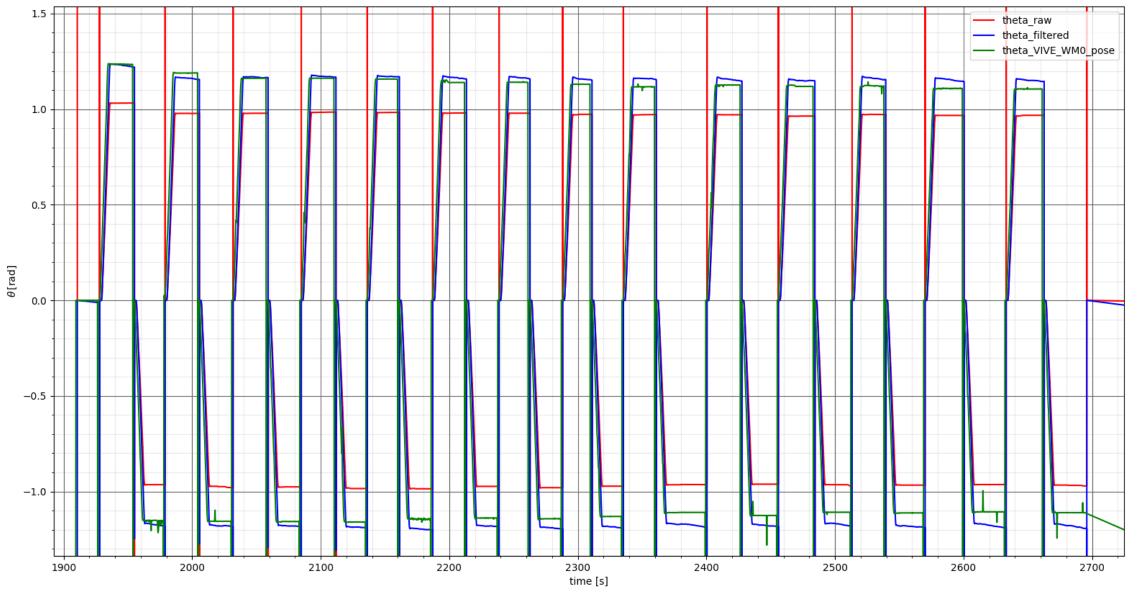


Figure 6.68: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving forward and backward (detail of the 3rd sub-interval between times of 1900 s and 2700 s; forward angular velocity of 0.2 rad/s)

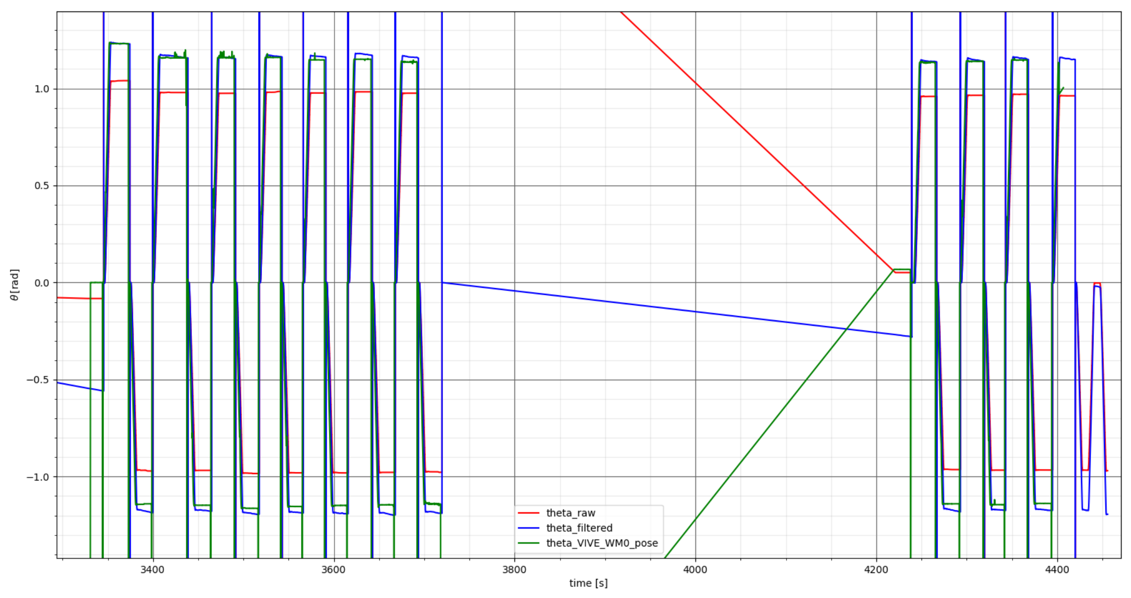


Figure 6.69: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving forward and backward (detail of the last 2 sub-intervals between times of 3300 s and 4500 s; forward angular velocity of 0.2 rad/s)

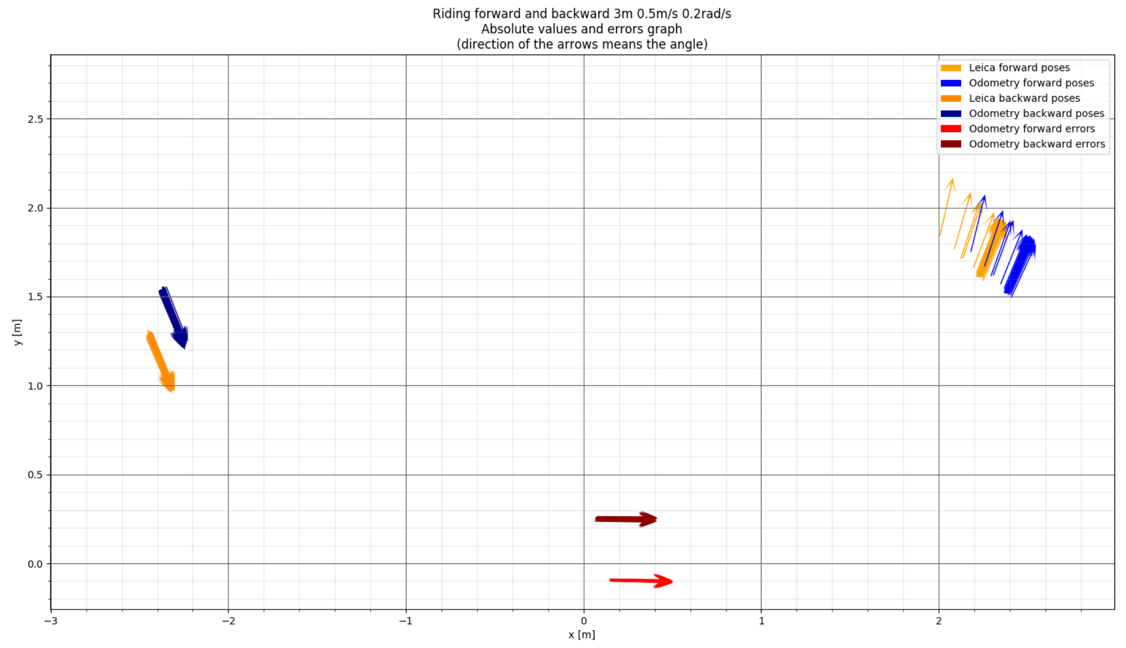


Figure 6.70: Plot of final poses and error vectors (towards the laser tracker final poses) of the robot's adjusted trajectories of the filtered odometry (forward angular velocity of 0.2 rad/s)

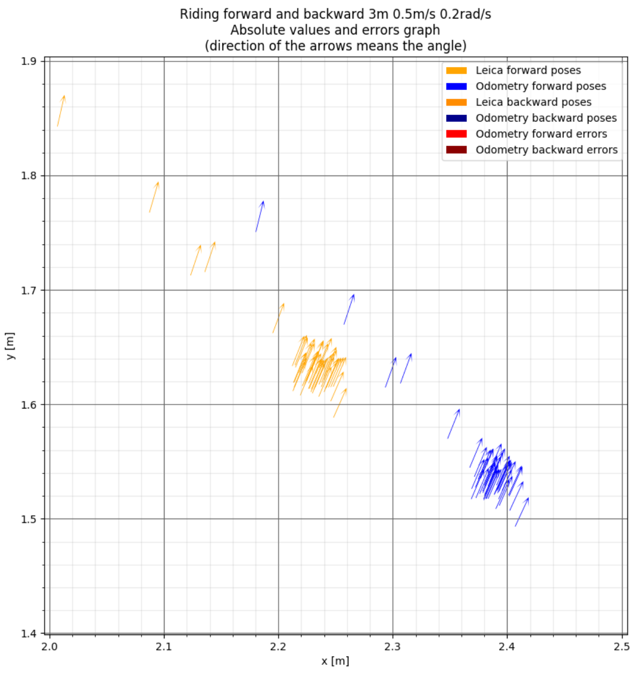


Figure 6.71: Plot of final poses of the robot's adjusted trajectories of the filtered odometry (forward angular velocity of 0.2 rad/s, forward detail)

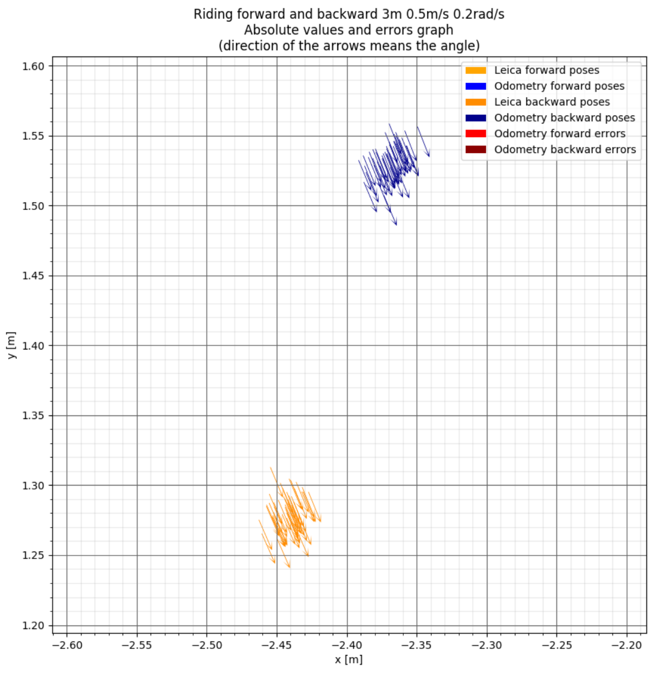


Figure 6.72: Plot of final poses of the robot's adjusted trajectories of the filtered odometry (forward angular velocity of 0.2 rad/s, backward detail)

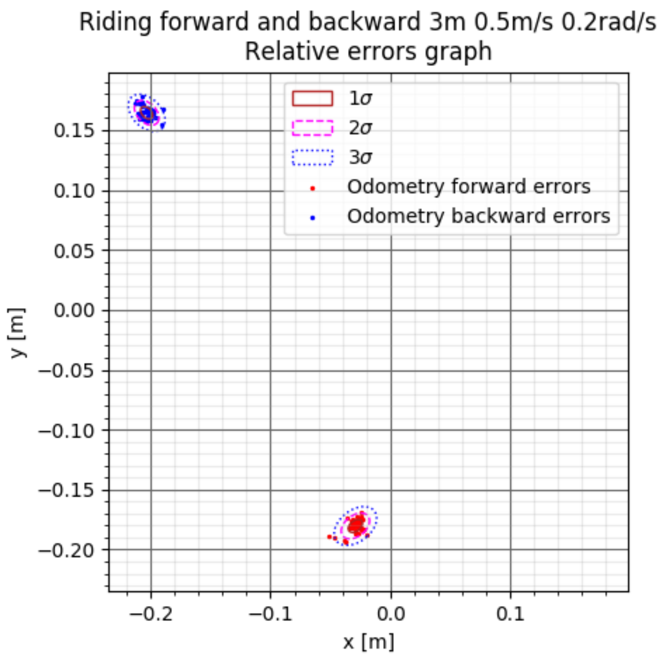


Figure 6.73: Plot of x and y components of relative errors of the final poses of the robot's adjusted trajectories of odometries (forward angular velocity of 0.2 rad/s)

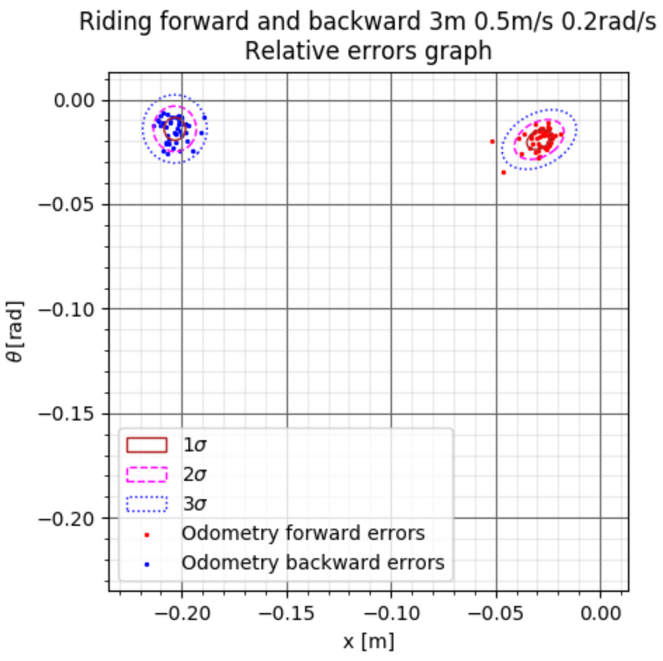


Figure 6.74: Plot of x and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (forward angular velocity of 0.2 rad/s)

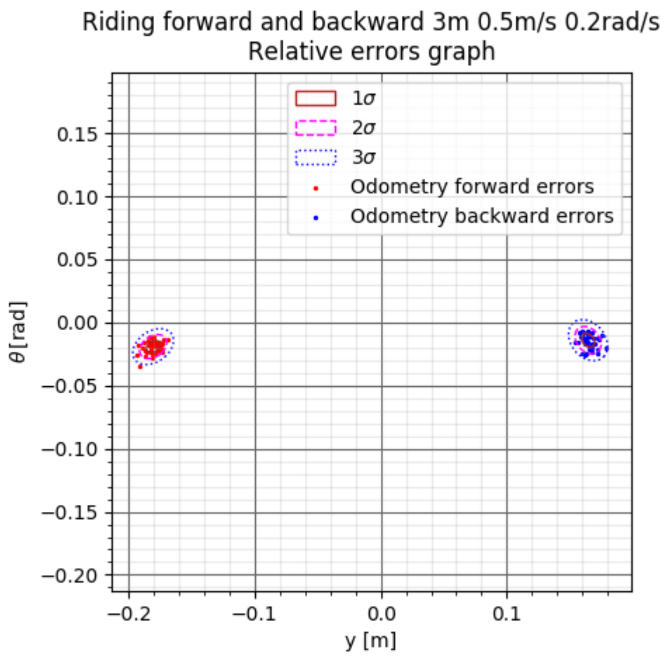


Figure 6.75: Plot of y and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (forward angular velocity of 0.2 rad/s)

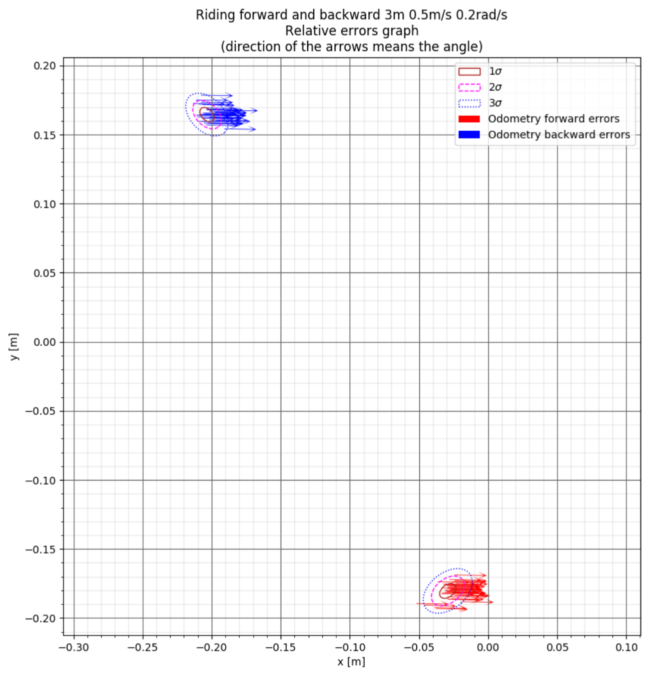


Figure 6.76: Plot of final error vectors of odometries towards the laser tracker reference data and their confidence ellipses (forward angular velocity of 0.2 rad/s)

Results of the experiment for this set of velocities. All the data from this part of the experiment, where the IMU had been properly initialized, have been measured and processed. We can see from the plots of relative error vectors in the figures 6.73 through 6.75 that **the relative error vector can be approximated by a 3-dimensional normal distribution** in the same way and from the same reason described at the end of the subsection about the part of this experiment driven with null angular velocity on the page 96. Therefore the probabilistic distribution of the EKF-filtered odometry of the relative error vector for this set of velocities during the forward movement

- $v_x = 0.5\,\mathrm{m/s}$
- $v_\theta = 0.2\,\mathrm{rad/s}$

with properly initialized IMU can be characterized by the following statistics:

1. **The forward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu}_{\boldsymbol{e}}(0.5, 0.1, \mathrm{true}) = \begin{bmatrix} -0.0291\,\mathrm{m} & -0.1804\,\mathrm{m} & -0.0192\,\mathrm{rad} \end{bmatrix}^T \tag{6.19}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{e}}(0.5, 0.1, \mathrm{true}) = \begin{bmatrix} 34.330 \cdot 10^{-6}\,\mathrm{m}^2 & 12.467 \cdot 10^{-6}\,\mathrm{m}^2 & 9.528 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} \\ 12.467 \cdot 10^{-6}\,\mathrm{m}^2 & 28.322 \cdot 10^{-6}\,\mathrm{m}^2 & 8.855 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} \\ 9.528 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} & 8.855 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} & 22.033 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.20}$$

2. **The backward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu}_{\boldsymbol{e}}(-0.5, -0.1, \mathrm{true}) = \begin{bmatrix} -0.2033\,\mathrm{m} & 0.1644\,\mathrm{m} & -0.0141\,\mathrm{rad} \end{bmatrix}^T \tag{6.21}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{e}}(-0.5, -0.1, \mathrm{true}) = \begin{bmatrix} 25.706 \cdot 10^{-6}\,\mathrm{m}^2 & -10.281 \cdot 10^{-6}\,\mathrm{m}^2 & -0.087 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} \\ -10.281 \cdot 10^{-6}\,\mathrm{m}^2 & 26.042 \cdot 10^{-6}\,\mathrm{m}^2 & -7.910 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} \\ -0.087 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} & -7.910 \cdot 10^{-6}\,\mathrm{rad} \cdot \mathrm{m} & 28.926 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.22}$$

Driving 3 meters forward and backward at 0.5 m/s with angular velocity of -0.2 rad/s

This trajectory was driven in 33 successfully recorded loops split into 2 whole intervals of 17 and 16, respectively, driven loops. It is apparent from the plots of θ-components of the odometries in the figures 6.82 and 6.97 that **2 different setups of the IMU appear in the data**. In the **first 17 loops, the IMU was improperly initialized**, whereas in the **last 16 loops, the IMU was not properly initialized**. It is also apparent from the following plots of the final positions and relative error vectors, respectively, of both intervals at once in the figures 6.77 and 6.78, respectively, where the vectors form rather 2 distinct significantly different Gaussians than 1 throughout the loops.

*The **plot of the robot's paths*** (at least their final poses) read from

1. Raw encoder odometry (only in some of the figures),
2. Filtered odometry *and*
3. the tracking system odometry (only in some of the figures),

adjusted always to the null position when changing its direction from forward to backward or vice versa (assumed just after having had its pose measured by the laser tracker), with the ***respective graphs of errors of the filtered odometry towards the laser tracker*** *can be seen*

1. in the figures 6.79 through 6.93 *with details for the **first** interval* of loops
2. in the figures 6.94 through 6.104 *with details for the **second** interval* of loops

The graphs show that **the robot keeps subtly steering left it the direction of its driving**, even though it is commanded to drive straight with null angular velocity. In reality it results in **permanently increasing tilt of the direction of the robot's straight trajectory counter-clockwise**. Also final poses estimated from the laser tracker measurement appear in some of the graphs (marked in the legends of the particular graphs).

The graphs show that **the robot keeps again subtly steering slightly more left it the direction of its driving**, in addition to the commanded angular velocity, which has the same consequences as during the previous parts of the experiment.

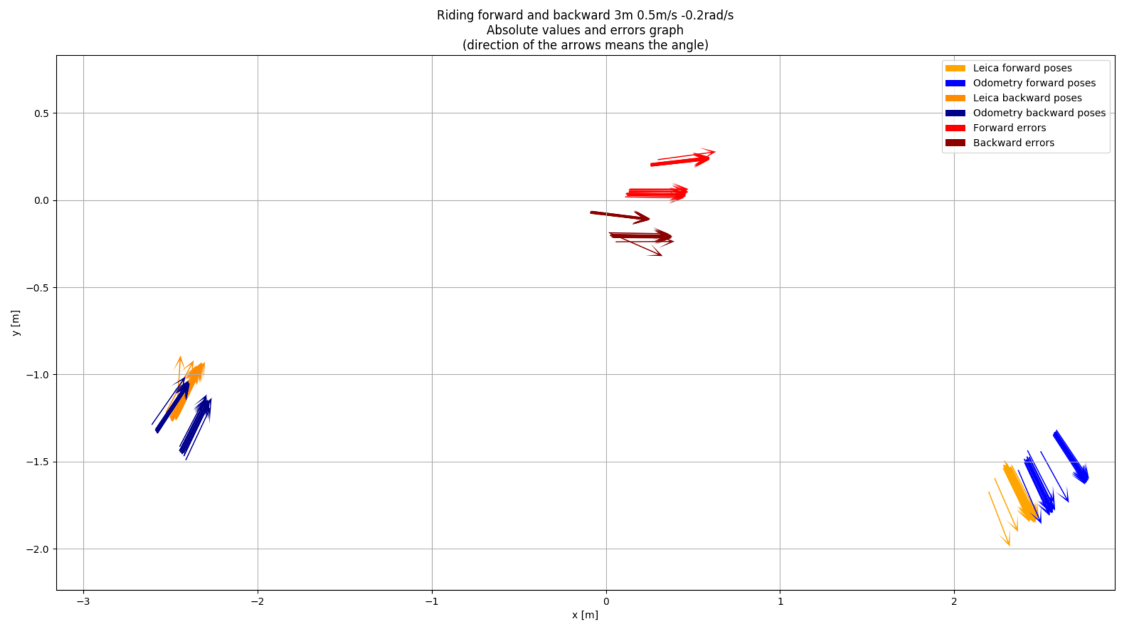


Figure 6.77: Plot of final poses and error vectors (towards the laser tracker final poses) of the robot's adjusted trajectories of the filtered odometry (both intervals; forward angular velocity of -0.2 rad/s)

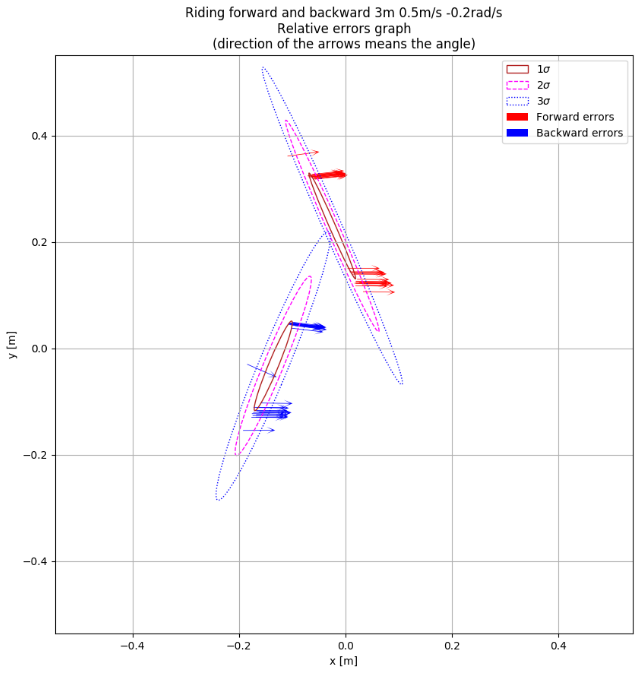


Figure 6.78: Plot of final error vectors of odometries towards the laser tracker reference data and their confidence ellipses (both intervals; forward angular velocity of -0.2 rad/s)

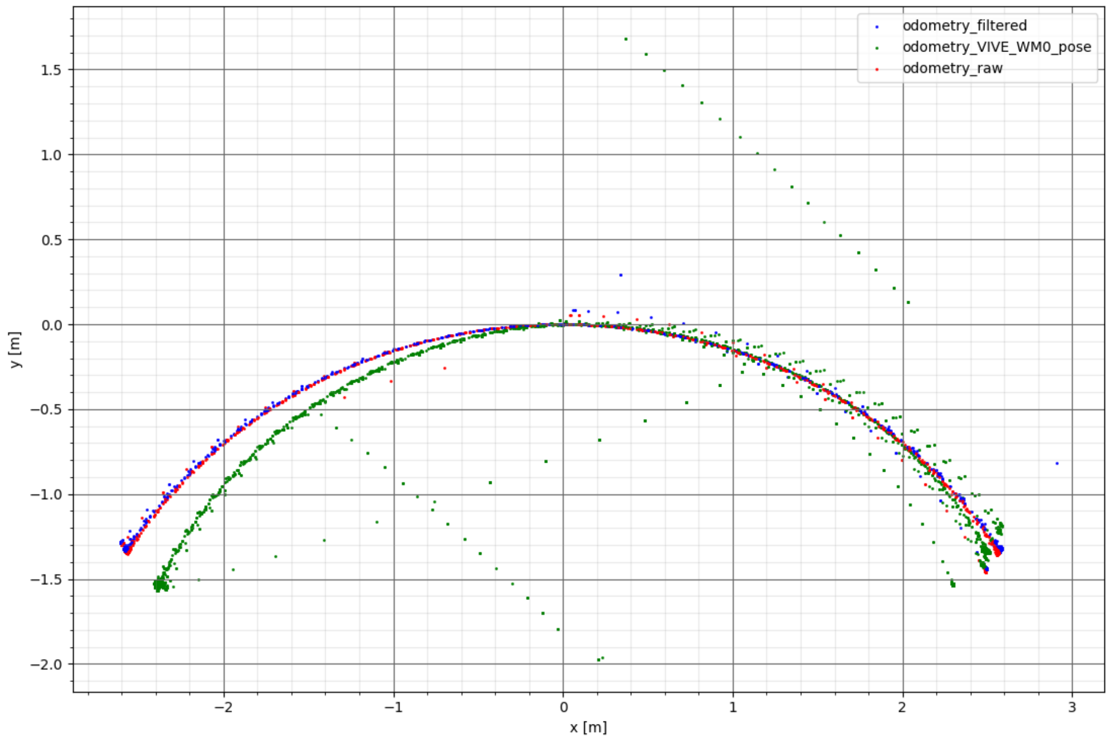


Figure 6.79: Plot of XY-data of the robot's adjusted trajectories when repeatedly driving forward and backward (first interval; forward angular velocity of -0.2 rad/s)

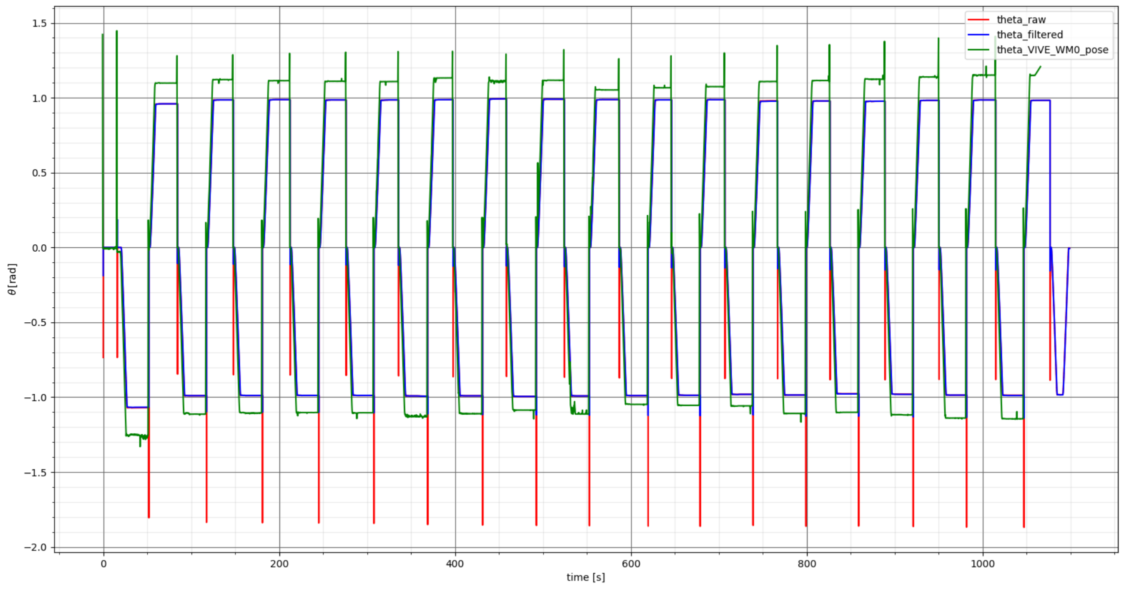


Figure 6.80: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving forward and backward (first interval; note that here the raw encoder odometry has even values with opposite signs; forward angular velocity of -0.2 rad/s)

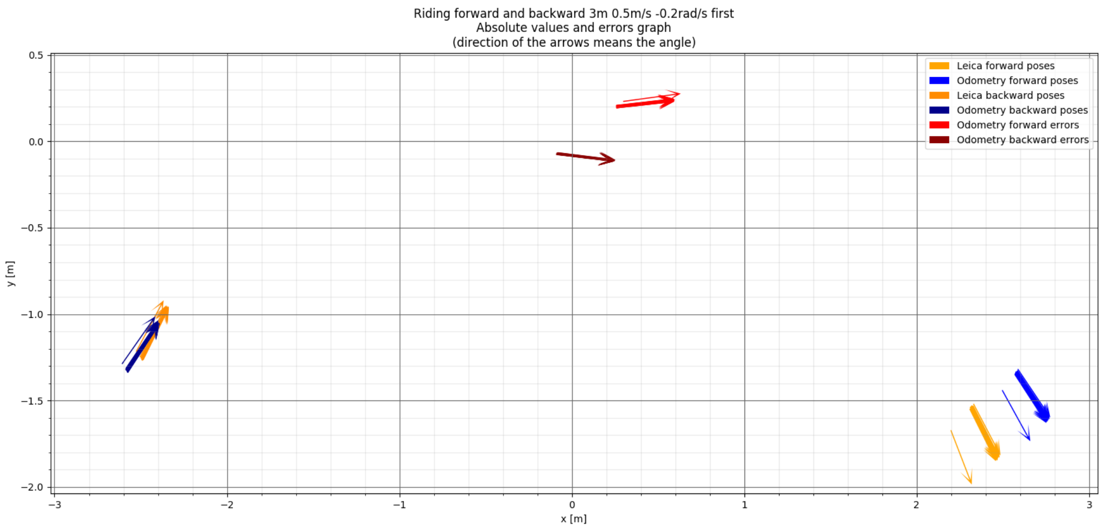


Figure 6.81: Plot of final poses and error vectors (towards the laser tracker final poses) of the robot's adjusted trajectories of the filtered odometry (first interval; forward angular velocity of −0.2 rad/s)

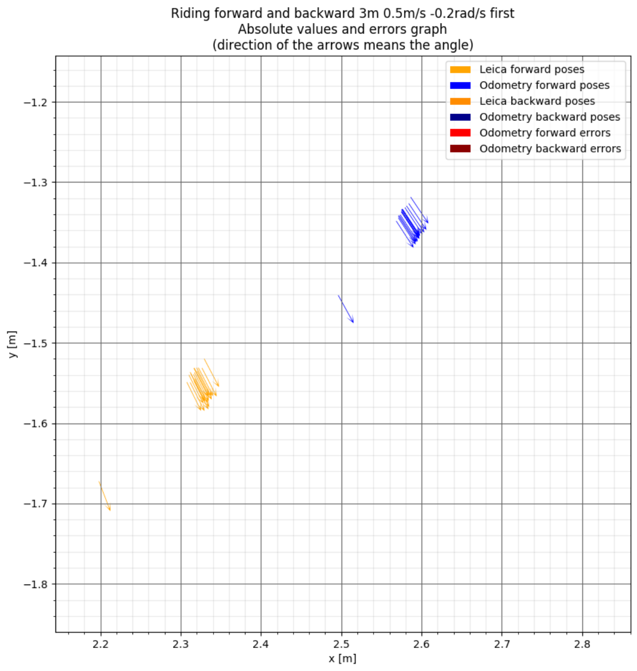


Figure 6.82: Plot of final poses of the robot's adjusted trajectories of the filtered odometry (first interval; forward angular velocity of −0.2 rad/s, forward detail)

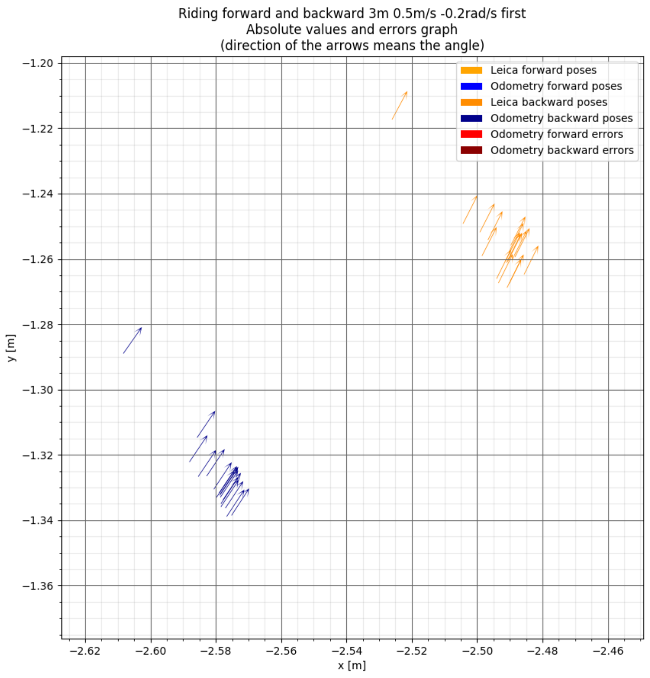


Figure 6.83: Plot of final poses of the robot's adjusted trajectories of the filtered odometry (first interval; forward angular velocity of -0.2 rad/s, backward detail)

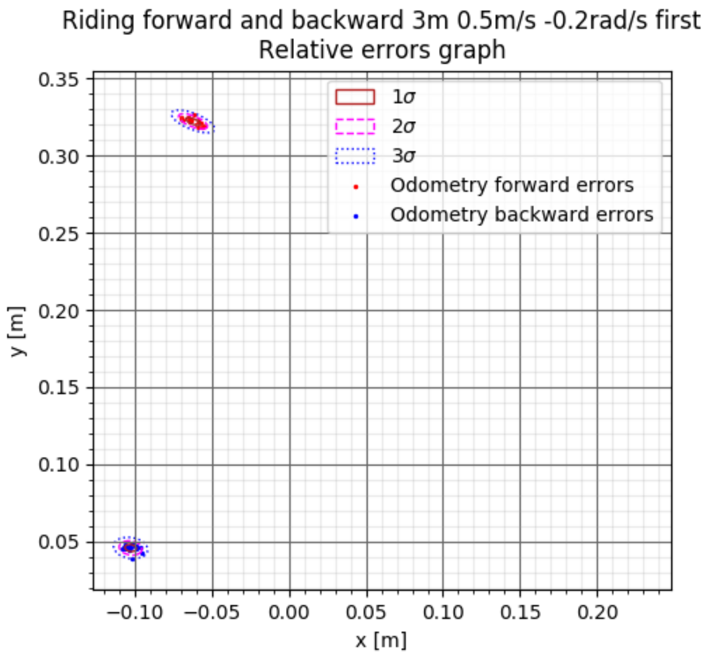


Figure 6.84: Plot of x and y components of relative errors of the final poses of the robot's adjusted trajectories of odometries (first interval; forward angular velocity of -0.2 rad/s)

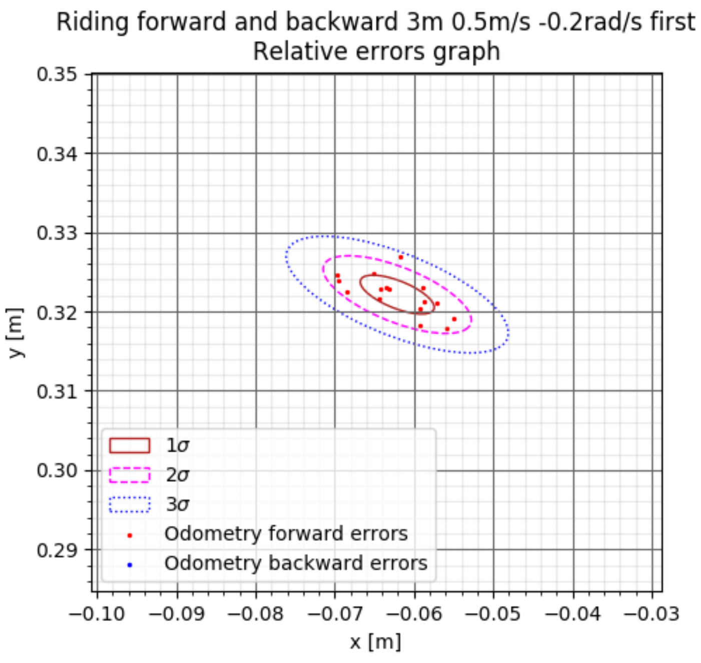


Figure 6.85: Plot of x and y components of relative errors of the final poses of the robot's adjusted trajectories of odometries (first interval; forward angular velocity of -0.2 rad/s, forward detail)

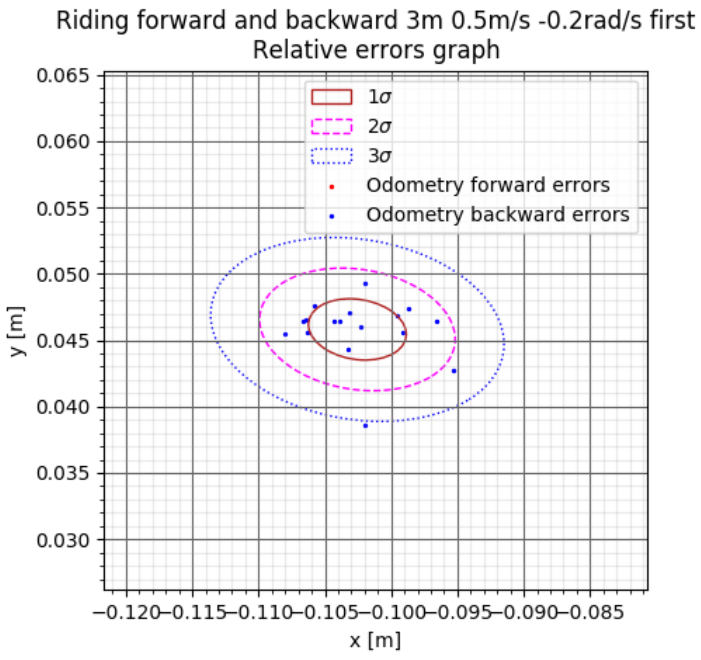


Figure 6.86: Plot of x and y components of relative errors of the final poses of the robot's adjusted trajectories of odometries (first interval; forward angular velocity of -0.2 rad/s, backward detail)

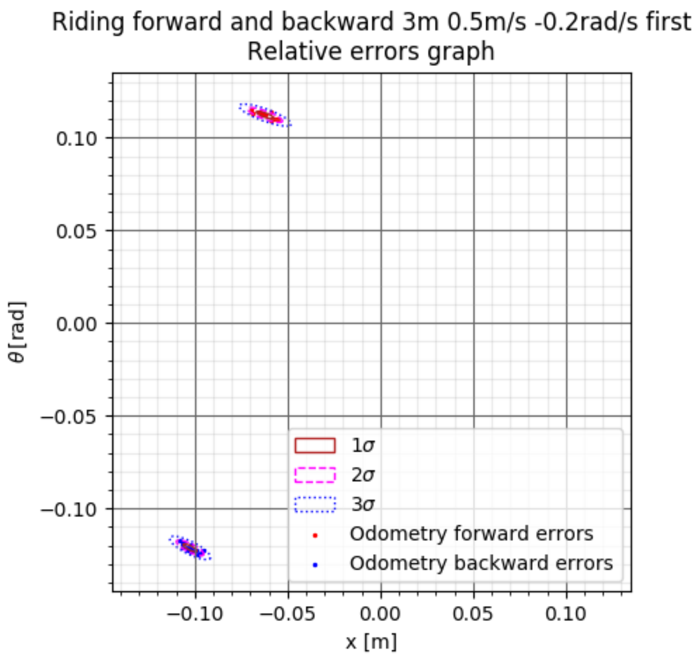


Figure 6.87: Plot of x and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (first interval; forward angular velocity of -0.2 rad/s)

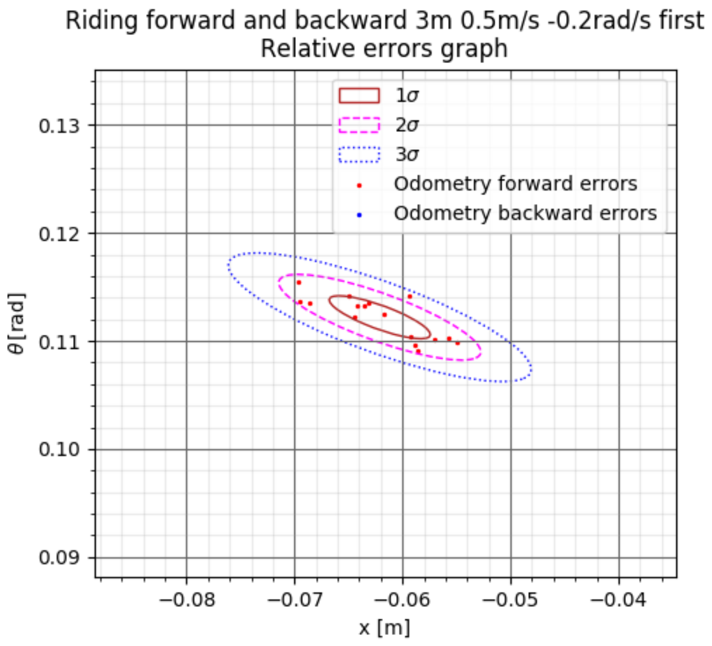


Figure 6.88: Plot of x and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (first interval; forward angular velocity of -0.2 rad/s, forward detail)

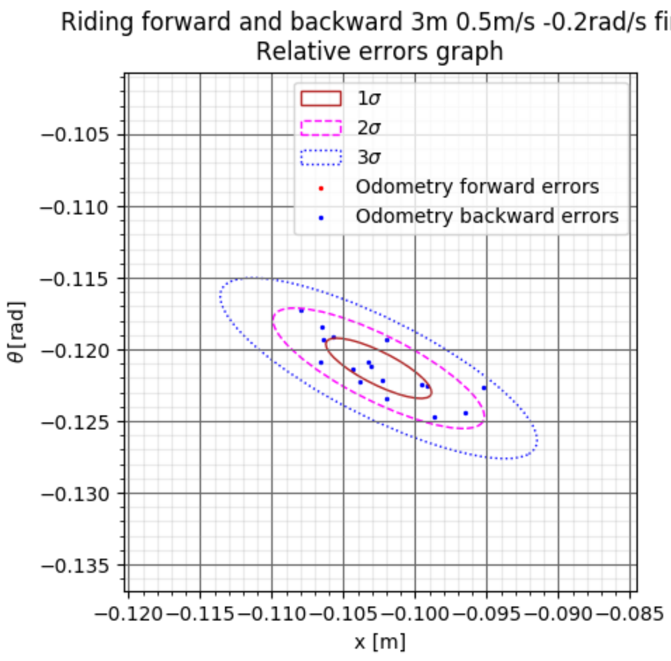


Figure 6.89: Plot of x and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (first interval; forward angular velocity of -0.2 rad/s, backward detail)

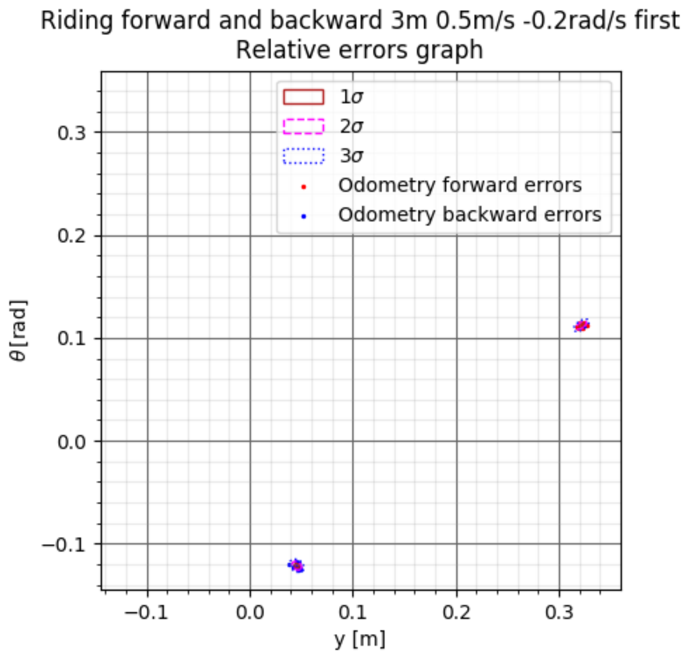


Figure 6.90: Plot of y and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (first interval; forward angular velocity of -0.2 rad/s)

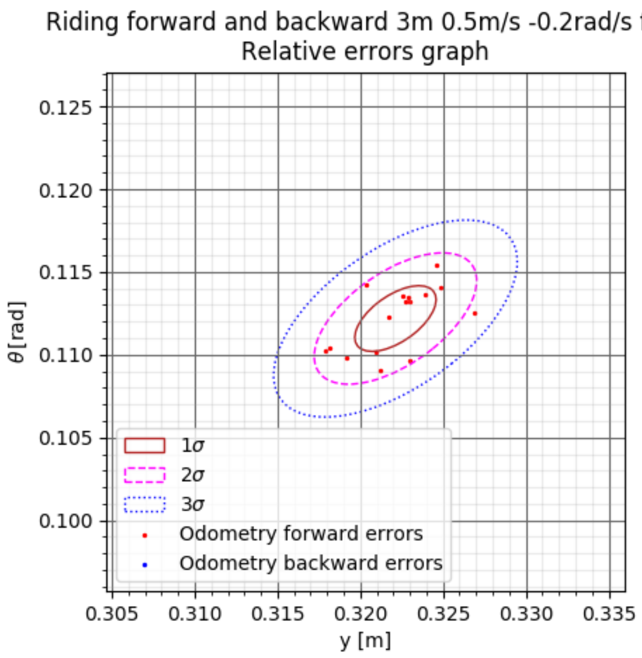


Figure 6.91: Plot of y and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (first interval; forward angular velocity of -0.2 rad/s, forward detail)

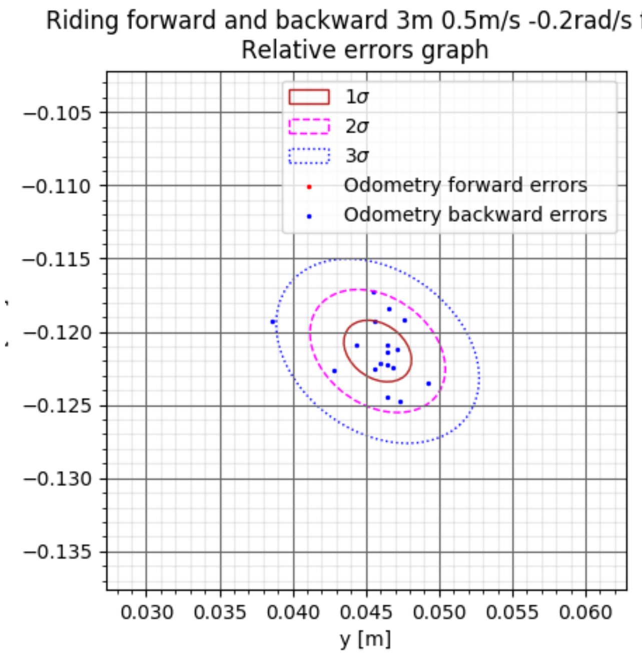


Figure 6.92: Plot of y and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (first interval; forward angular velocity of -0.2 rad/s, backward detail)

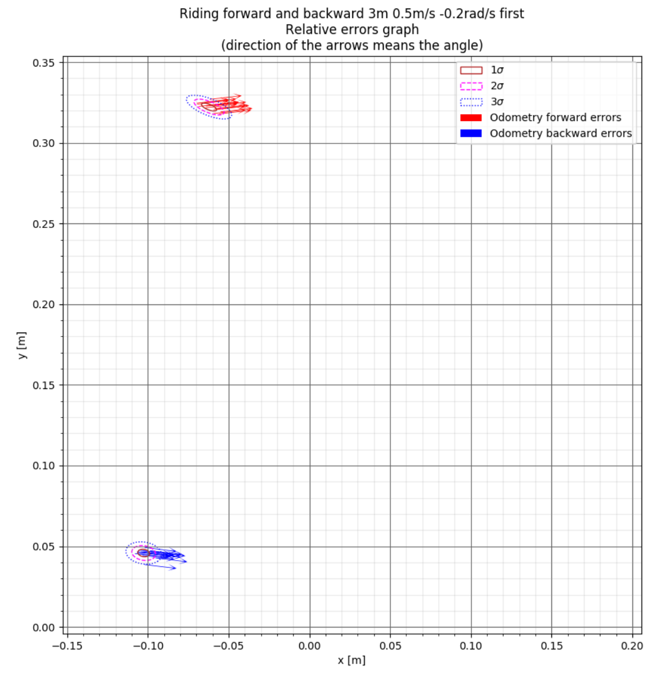


Figure 6.93: Plot of final error vectors of the filtered odometry towards the laser tracker reference data and their confidence ellipses (first interval; forward angular velocity of -0.2 rad/s)

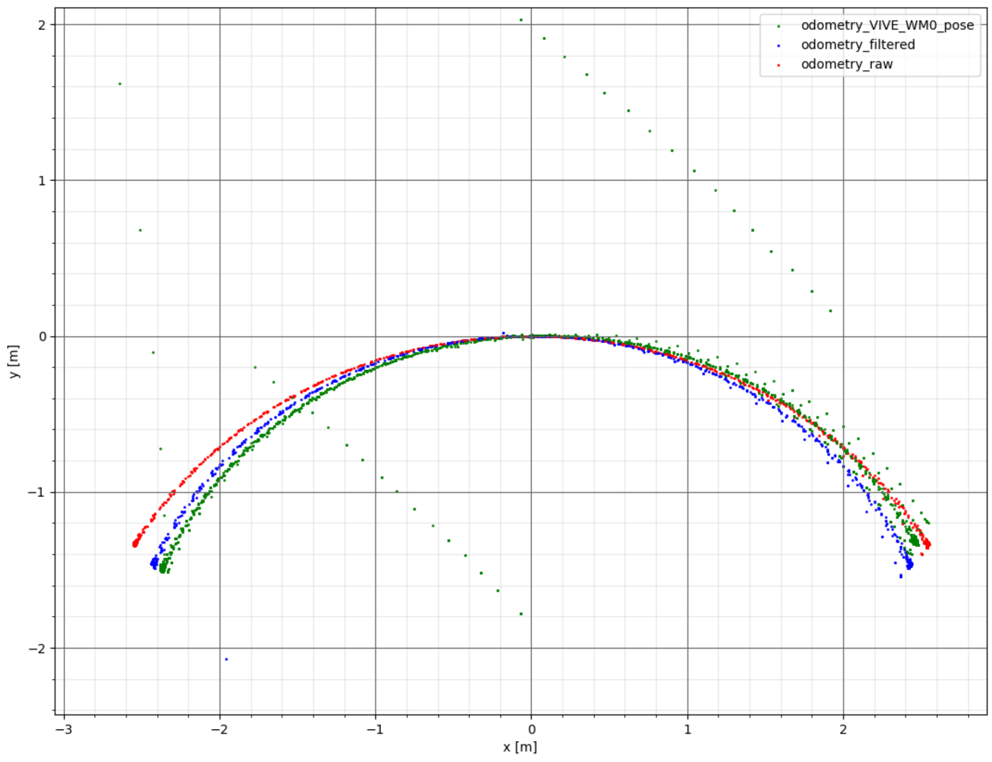


Figure 6.94: Plot of XY-data of the robot's adjusted trajectories when repeatedly driving forward and backward (second interval; forward angular velocity of -0.2 rad/s)

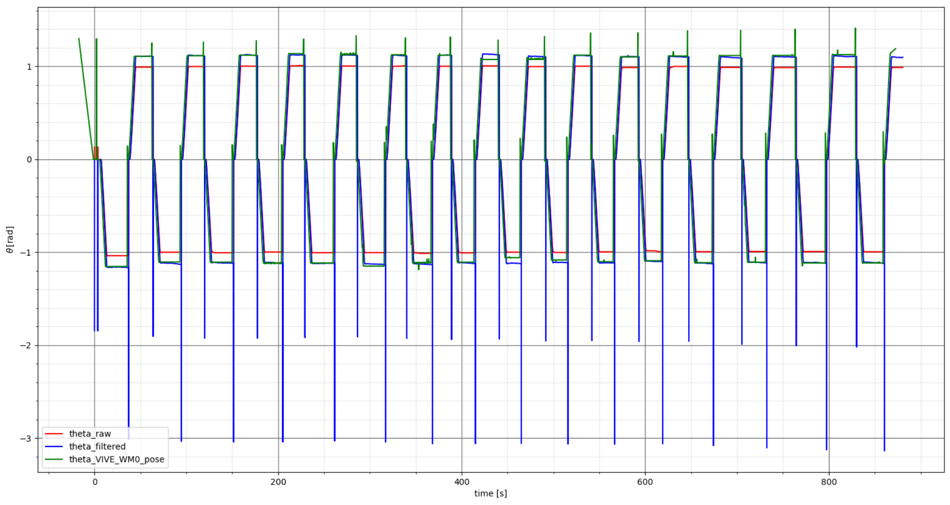


Figure 6.95: Plot of θ-data of the robot's adjusted trajectories when repeatedly driving forward and backward (second interval; forward angular velocity of -0.2 rad/s; note that here, the filtered odometry has the same values as the raw encoder odometry)

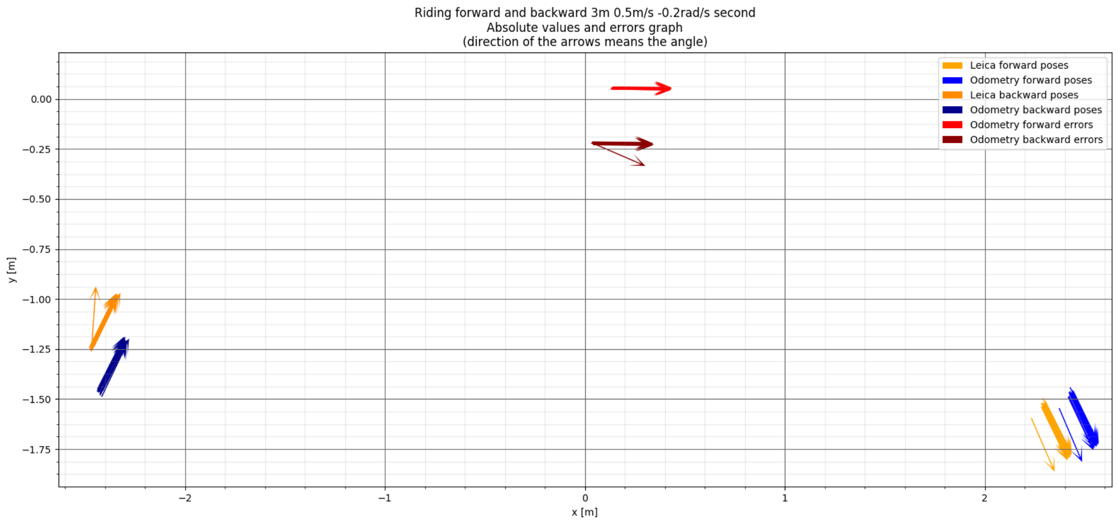


Figure 6.96: Plot of final poses and error vectors (towards the laser tracker final poses) of the robot's adjusted trajectories of the filtered odometry (second interval; forward angular velocity of -0.2 rad/s)

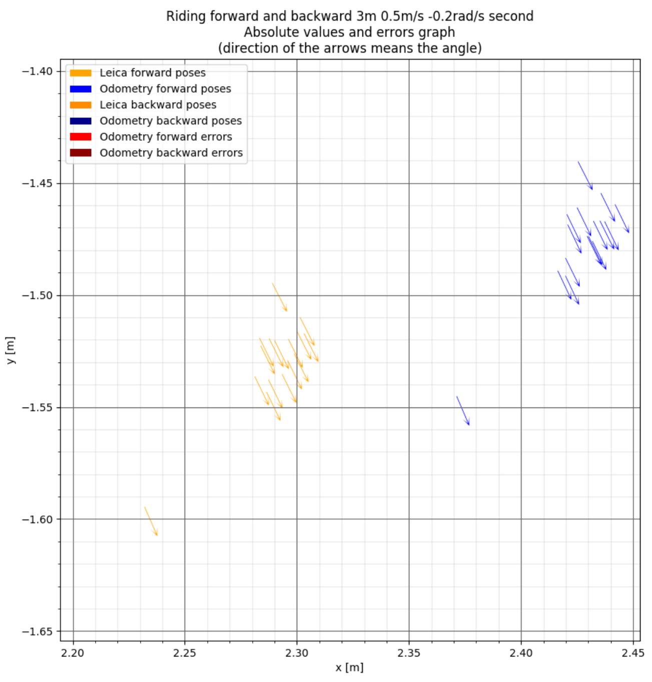


Figure 6.97: Plot of final poses of the robot's adjusted trajectories of the filtered odometry (second interval; forward angular velocity of −0.2 rad/s, forward detail)

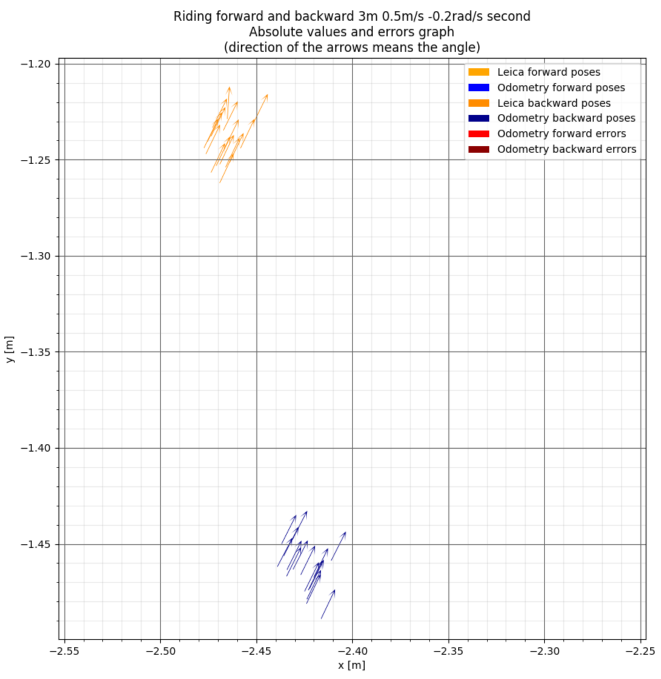


Figure 6.98: Plot of final poses of the robot's adjusted trajectories of the filtered odometry (second interval; forward angular velocity of −0.2 rad/s, backward detail)

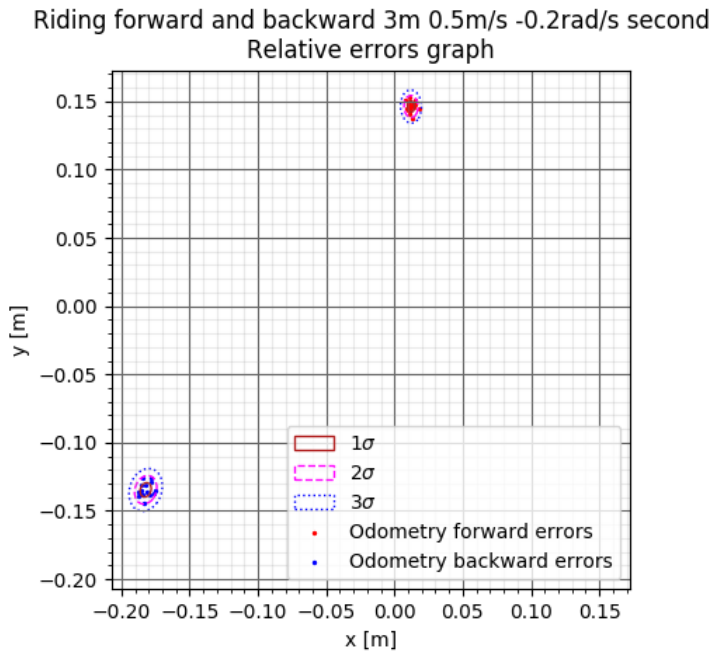


Figure 6.99: Plot of x and y components of relative errors of the final poses of the robot's adjusted trajectories of odometries (second interval; forward angular velocity of -0.2 rad/s)

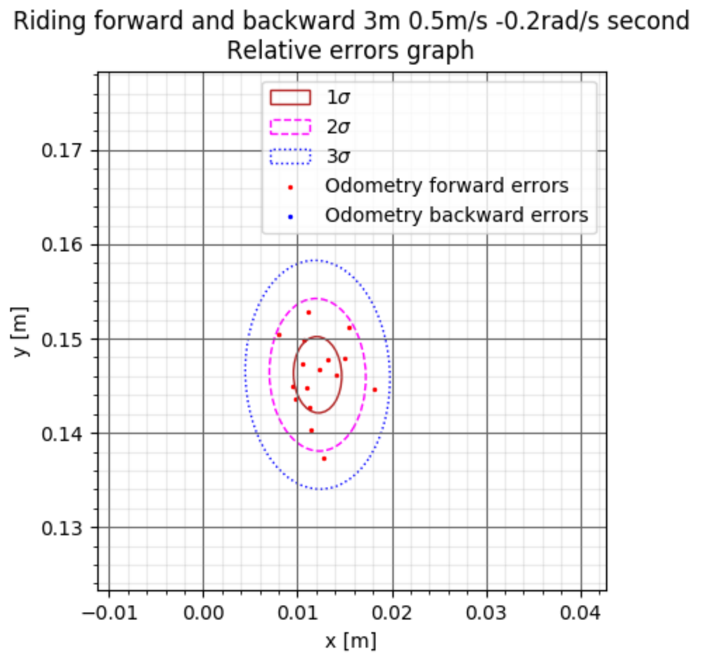


Figure 6.100: Plot of x and y components of relative errors of the final poses of the robot's adjusted trajectories of odometries (second interval; forward angular velocity of -0.2 rad/s, forward detail)

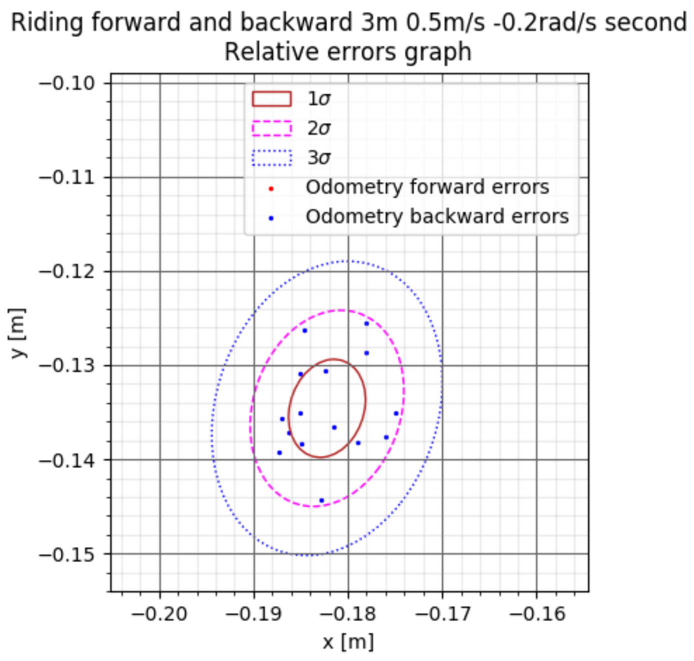


Figure 6.101: Plot of x and y components of relative errors of the final poses of the robot's adjusted trajectories of odometries (second interval; forward angular velocity of -0.2 rad/s, backward detail)

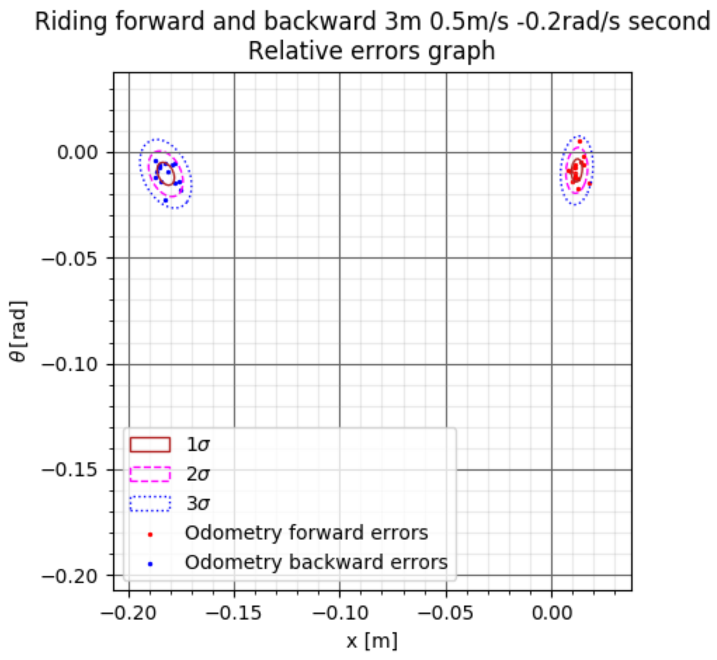


Figure 6.102: Plot of x and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (second interval; forward angular velocity of -0.2 rad/s)

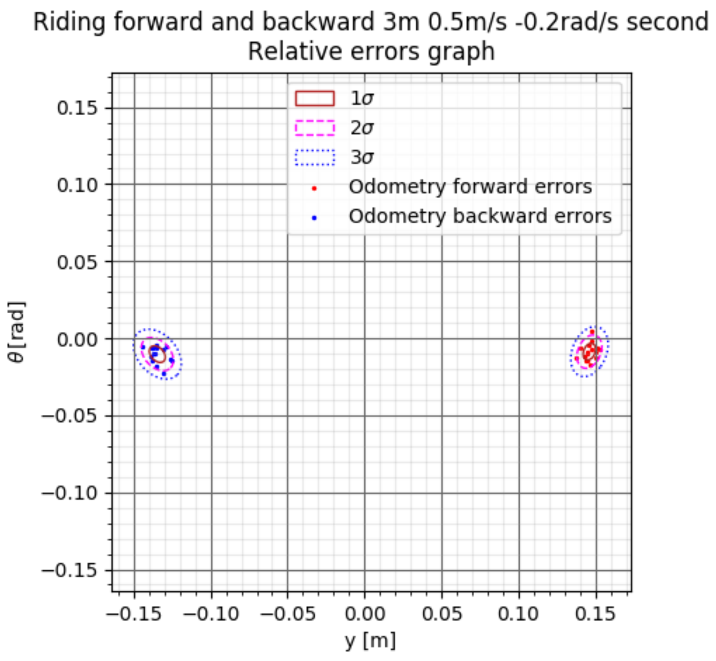


Figure 6.103: Plot of y and θ components of relative errors of the final poses of the robot's adjusted trajectories of odometries (second interval; forward angular velocity of -0.2 rad/s)

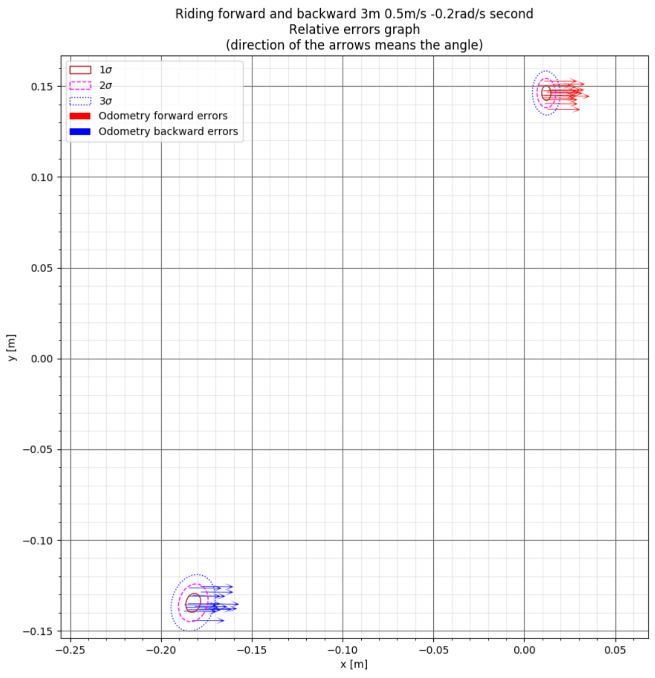


Figure 6.104: Plot of final error vectors of the filtered odometry towards the laser tracker reference data and their confidence ellipses (second interval; forward angular velocity of -0.2 rad/s)

Results from the part with improperly initialized IMU. All the data from the part of the experiment, where the IMU had not been properly initialized and thus had led to larger error vectors, have been measured and processed. We can see from the plots of relative error vectors in the figures 6.84 through 6.92 that **the relative error vector can be approximated by a 3-dimensional normal distribution** in the same way and from the same reason described at the end of the subsection about the part of this experiment driven with null angular velocity on the page 96. Therefore the probabilistic distribution of the EKF-filtered odometry of the relative error vector for this set of velocities during the forward movement

- $v_x = 0.5\,\mathrm{m/s}$
- $v_\theta = -0.2\,\mathrm{rad/s}$

with improperly initialized IMU corresponding to the first interval can be characterized by the following statistics:

1. **The forward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu_e}(0.5, -0.2, \mathrm{false}) = \begin{bmatrix} -0.0621\,\mathrm{m} & 0.3221\,\mathrm{m} & 0.1122\,\mathrm{rad} \end{bmatrix}^T \tag{6.23}$$

$$\boldsymbol{\Sigma_e}(0.5, -0.2, \mathrm{false}) = \begin{bmatrix} 20.490 \cdot 10^{-6}\,\mathrm{m}^2 & -7.057 \cdot 10^{-6}\,\mathrm{m}^2 & -7.019 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ -7.057 \cdot 10^{-6}\,\mathrm{m}^2 & 5.639 \cdot 10^{-6}\,\mathrm{m}^2 & 2.637 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ -7.019 \cdot 10^{-6}\,\mathrm{rad \cdot m} & 2.637 \cdot 10^{-6}\,\mathrm{rad \cdot m} & 3.685 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.24}$$

2. **The backward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu_e}(-0.5, 0.2, \mathrm{false}) = \begin{bmatrix} -0.1025\,\mathrm{m} & 0.0458\,\mathrm{m} & -0.1213\,\mathrm{rad} \end{bmatrix}^T \tag{6.25}$$

$$\boldsymbol{\Sigma_e}(-0.5, 0.2, \mathrm{false}) = \begin{bmatrix} 12.788 \cdot 10^{-6}\,\mathrm{m}^2 & -1.282 \cdot 10^{-6}\,\mathrm{m}^2 & -5.811 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ -1.282 \cdot 10^{-6}\,\mathrm{m}^2 & 5.022 \cdot 10^{-6}\,\mathrm{m}^2 & -1.341 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ -5.811 \cdot 10^{-6}\,\mathrm{rad \cdot m} & -1.341 \cdot 10^{-6}\,\mathrm{rad \cdot m} & 4.152 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.26}$$

These results correspond to the data after having removed **1 outlier measured in the forward direction**.

Results from the part with properly initialized IMU. All the data from the part of the experiment, where the IMU had been properly initialized and thus had led to smaller error vectors, have been measured and processed. We can see from the plots of relative error vectors in the figures 6.99 through 6.103 that **the relative error vector can be approximated by a 3-dimensional normal distribution** in the same way and from the same reason described at the end of the subsection about the part of this experiment driven with null angular velocity on the page 96. Therefore the probabilistic distribution of the EKF-filtered odometry of the relative error vector for this set of velocities during the forward movement

- $v_x = 0.5\,\mathrm{m/s}$
- $v_\theta = -0.2\,\mathrm{rad/s}$

with properly initialized IMU corresponding to the second interval can be characterized by the following statistics:

1. **The forward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu_e}(0.5, -0.2, \mathrm{true}) = \begin{bmatrix} 0.0121\,\mathrm{m} & 0.1462\,\mathrm{m} & -0.0087\,\mathrm{rad} \end{bmatrix}^T \tag{6.27}$$

$$\boldsymbol{\Sigma_e}(0.5, -0.2, \mathrm{true}) = \begin{bmatrix} 6.131 \cdot 10^{-6}\,\mathrm{m}^2 & -0.352 \cdot 10^{-6}\,\mathrm{m}^2 & 1.740 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ -0.352 \cdot 10^{-6}\,\mathrm{m}^2 & 15.345 \cdot 10^{-6}\,\mathrm{m}^2 & 5.881 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ 1.740 \cdot 10^{-6}\,\mathrm{rad \cdot m} & 5.881 \cdot 10^{-6}\,\mathrm{rad \cdot m} & 27.326 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.28}$$

2. **The backward movement errors mean vector and covariance matrix:**

$$\boldsymbol{\mu_e}(-0.5, 0.2, \mathrm{true}) = \begin{bmatrix} -0.1822\,\mathrm{m} & -0.13461\,\mathrm{m} & 0.0052\,\mathrm{rad} \end{bmatrix}^T \tag{6.29}$$

$$\boldsymbol{\Sigma_e}(-0.5, 0.2, \mathrm{true}) = \begin{bmatrix} 15.481 \cdot 10^{-6}\,\mathrm{m}^2 & 3.550 \cdot 10^{-6}\,\mathrm{m}^2 & -7.233 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ 3.550 \cdot 10^{-6}\,\mathrm{m}^2 & 25.236 \cdot 10^{-6}\,\mathrm{m}^2 & -9.690 \cdot 10^{-6}\,\mathrm{rad \cdot m} \\ -7.233 \cdot 10^{-6}\,\mathrm{rad \cdot m} & -9.690 \cdot 10^{-6}\,\mathrm{rad \cdot m} & 27.157 \cdot 10^{-6}\,\mathrm{rad}^2 \end{bmatrix} \tag{6.30}$$

These results correspond to the data after having removed **1 outlier measured in the backward direction**.

6.4.4 Results of the whole experiment

The mean error vectors and error covariance matrices have been computed for totally 8 pairs of velocities (4 for the positive and 4 for the negative forward velocity):

- $v_x = 0.5\,\mathrm{m/s}$ and $v_\theta \in \{0, 0.1, 0.2, -0.2\}$ rad/s
- $v_x = -0.5\,\mathrm{m/s}$ and $v_\theta \in \{0, -0.1, -0.2, 0.2\}$ rad/s

Having used the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ expressed in (4.64) and (4.65), 2 different matrices of quadratic coefficients expressed by (4.66) have been approximated for the positive and for the negative forward velocity (± 0.5 m/s) by the least squares method. They approximate the quadratic dependency of the values of components of the mean error vector and error covariance matrix on the commanded angular velocity, all that per 3 meters driven.

Assume that e_Y means a sum of squared errors of the approximated model from the actual instances of parameter Y, where parameter Y means any of the parameters described in the term (4.64). Then the resulting approximated matrices can be approximated as follows:

- For the forward velocity $v_x = 0.5\,\mathrm{m/s}$ the resulting matrix of coefficients expressed by (4.66) (transposed here for better notation) is:

$$\boldsymbol{X}\,(0.5\,\mathrm{m/s}) = \begin{bmatrix} a_{e_x} & b_{e_x} & c_{e_x} \\ a_{e_y} & b_{e_y} & c_{e_y} \\ a_{e_\theta} & b_{e_\theta} & c_{e_\theta} \\ a_{m_1} & b_{m_1} & c_{m_1} \\ a_{m_2} & b_{m_2} & c_{m_2} \\ a_{m_3} & b_{m_3} & c_{m_3} \\ a_{m_4} & b_{m_4} & c_{m_4} \\ a_{m_5} & b_{m_5} & c_{m_5} \\ a_{m_6} & b_{m_6} & c_{m_6} \end{bmatrix}^T$$

$$\doteq \begin{bmatrix} -1.4798\ \frac{\mathrm{m \cdot s^2}}{\mathrm{rad^2}} & -0.0983\ \frac{\mathrm{m \cdot s}}{\mathrm{rad}} & 0.0512\ \mathrm{m} \\ -0.1538\ \frac{\mathrm{m \cdot s^2}}{\mathrm{rad^2}} & -0.8210\ \frac{\mathrm{m \cdot s}}{\mathrm{rad}} & -0.0114\ \mathrm{m} \\ -0.1099\ \frac{\mathrm{s^2}}{\mathrm{rad}} & -0.0311\ \mathrm{s} & -0.0101\ \mathrm{rad} \\ 345.325 \cdot 10^{-6}\ \frac{\mathrm{m^2 s^2}}{\mathrm{rad^2}} & 62.251 \cdot 10^{-6}\ \frac{\mathrm{m^2 s}}{\mathrm{rad}} & 5.593 \cdot 10^{-6}\ \mathrm{m^2} \\ 177.028 \cdot 10^{-6}\ \frac{\mathrm{m^2 s^2}}{\mathrm{rad^2}} & 33.620 \cdot 10^{-6}\ \frac{\mathrm{m^2 s}}{\mathrm{rad}} & -0.867 \cdot 10^{-6}\ \mathrm{m^2} \\ 197.523 \cdot 10^{-6}\ \frac{\mathrm{m \cdot s^2}}{\mathrm{rad}} & 20.892 \cdot 10^{-6}\ \mathrm{m \cdot s} & -2.125 \cdot 10^{-6}\ \mathrm{rad \cdot m} \\ -173.266 \cdot 10^{-6}\ \frac{\mathrm{m^2 s^2}}{\mathrm{rad^2}} & 33.546 \cdot 10^{-6}\ \frac{\mathrm{m^2 s}}{\mathrm{rad}} & 28.874 \cdot 10^{-6}\ \mathrm{m^2} \\ -171.513 \cdot 10^{-6}\ \frac{\mathrm{m \cdot s^2}}{\mathrm{rad}} & 8.618 \cdot 10^{-6}\ \mathrm{m \cdot s} & 14.279 \cdot 10^{-6}\ \mathrm{rad \cdot m} \\ -899.730 \cdot 10^{-6}\ \mathrm{s^2} & 3.848 \cdot 10^{-6}\ \mathrm{rad \cdot s} & 29.987 \cdot 10^{-6}\ \mathrm{rad^2} \end{bmatrix}^T \tag{6.31}$$

with the corresponding vector of residuals

$$\boldsymbol{e} = \begin{bmatrix} e_{e_x} \\ e_{e_y} \\ e_{e_\theta} \\ e_{m_1} \\ e_{m_2} \\ e_{m_3} \\ e_{m_4} \\ e_{m_5} \\ e_{m_6} \end{bmatrix} \doteq \begin{bmatrix} 24.128 \cdot 10^{-6}\ \mathrm{m^2} \\ 24.367 \cdot 10^{-6}\ \mathrm{m^2} \\ 28.067 \cdot 10^{-6}\ \mathrm{rad^2} \\ 74.785 \cdot 10^{-12}\ \mathrm{m^4} \\ 2.719 \cdot 10^{-12}\ \mathrm{m^4} \\ 2.226 \cdot 10^{-12}\ \mathrm{rad^2 m^2} \\ 1.338 \cdot 10^{-12}\ \mathrm{m^4} \\ 3.801 \cdot 10^{-12}\ \mathrm{rad^2 m^2} \\ 321.006 \cdot 10^{-12}\ \mathrm{rad^4} \end{bmatrix} \tag{6.32}$$

- For the forward velocity $v_x = -0.5\,\mathrm{m/s}$ the resulting matrix of coefficients expressed by (4.66) (transposed here for better notation) is:

$$\boldsymbol{X}\,(-0.5\,\mathrm{m/s}) = \begin{bmatrix} a_{e_x} & b_{e_x} & c_{e_x} \\ a_{e_y} & b_{e_y} & c_{e_y} \\ a_{e_\theta} & b_{e_\theta} & c_{e_\theta} \\ a_{m_1} & b_{m_1} & c_{m_1} \\ a_{m_2} & b_{m_2} & c_{m_2} \\ a_{m_3} & b_{m_3} & c_{m_3} \\ a_{m_4} & b_{m_4} & c_{m_4} \\ a_{m_5} & b_{m_5} & c_{m_5} \\ a_{m_6} & b_{m_6} & c_{m_6} \end{bmatrix}^T$$

$$\doteq \begin{bmatrix} -3.6423\ \frac{\mathrm{m\cdot s^2}}{\mathrm{rad^2}} & 0.0483\ \frac{\mathrm{m\cdot s}}{\mathrm{rad}} & -0.0466\ \mathrm{m} \\ 0.0036\ \frac{\mathrm{m\cdot s^2}}{\mathrm{rad^2}} & -0.7654\ \frac{\mathrm{m\cdot s}}{\mathrm{rad}} & -0.0166\ \mathrm{m} \\ -0.0816\ \frac{\mathrm{s^2}}{\mathrm{rad}} & 0.0154\ \mathrm{s} & -0.0096\ \mathrm{rad} \\ 184.040\cdot 10^{-6}\ \frac{\mathrm{m^2 s^2}}{\mathrm{rad^2}} & -22.959\cdot 10^{-6}\ \frac{\mathrm{m^2 s}}{\mathrm{rad}} & 12.971\cdot 10^{-6}\ \mathrm{m^2} \\ -22.473\cdot 10^{-6}\ \frac{\mathrm{m^2 s^2}}{\mathrm{rad^2}} & 32.546\cdot 10^{-6}\ \frac{\mathrm{m^2 s}}{\mathrm{rad}} & -2.264\cdot 10^{-6}\ \mathrm{m^2} \\ -312.181\cdot 10^{-6}\ \frac{\mathrm{m\cdot s^2}}{\mathrm{rad}} & -13.624\cdot 10^{-6}\ \mathrm{m\cdot s} & 8.403\cdot 10^{-6}\ \mathrm{rad\cdot m} \\ 312.538\cdot 10^{-6}\ \frac{\mathrm{m^2 s^2}}{\mathrm{rad^2}} & -1.896\cdot 10^{-6}\ \frac{\mathrm{m^2 s}}{\mathrm{rad}} & 13.125\cdot 10^{-6}\ \mathrm{m^2} \\ 13.634\cdot 10^{-6}\ \frac{\mathrm{m\cdot s^2}}{\mathrm{rad}} & -7.166\cdot 10^{-6}\ \mathrm{m\cdot s} & -9.074\cdot 10^{-6}\ \mathrm{rad\cdot m} \\ 77.413\cdot 10^{-6}\ \mathrm{s^2} & 3.734\cdot 10^{-6}\ \mathrm{rad\cdot s} & 24.129\cdot 10^{-6}\ \mathrm{rad^2} \end{bmatrix}^T \tag{6.33}$$

with the corresponding vector of residuals

$$\boldsymbol{e} = \begin{bmatrix} e_{e_x} \\ e_{e_y} \\ e_{e_\theta} \\ e_{m_1} \\ e_{m_2} \\ e_{m_3} \\ e_{m_4} \\ e_{m_5} \\ e_{m_6} \end{bmatrix} = \begin{bmatrix} 22.608\cdot 10^{-6}\ \mathrm{m^2} \\ 347.429\cdot 10^{-6}\ \mathrm{m^2} \\ 40.689\cdot 10^{-6}\ \mathrm{rad^2} \\ 7.454\cdot 10^{-12}\ \mathrm{m^4} \\ 4.543\cdot 10^{-12}\ \mathrm{m^4} \\ 19.788\cdot 10^{-12}\ \mathrm{rad^2 m^2} \\ 0.016\cdot 10^{-12}\ \mathrm{m^4} \\ 8.110\cdot 10^{-12}\ \mathrm{rad^2 m^2} \\ 73.170\cdot 10^{-12}\ \mathrm{rad^4} \end{bmatrix} \tag{6.34}$$

It has been verified that this result is computed numerically stably by computing also the same quadratic coefficients yet separated for the values of the error mean vector and of the vectorized upper diagonal block of the covariance matrix, which yielded the same results (even also not rounded unlike the results here). It implies that the used method is numerically stable for the varying decadical orders of values contained in both the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

If we consider that the residuals are all sums of 4 instances of squared errors for a driven distance of 3 meters, the actual values of the residuals of each of the approximated parameter dependencies show that the computed quadratic dependency approximation can be used in order to determine the error mean vector and the covariance matrix during the robot's drive. Indeed, even the largest of the values of residuals of $e_{e_y} = 347.429 \cdot 10^{-6}$ m^{2}divided by 4 (as the count of summed squared errors), square-rooted and divided by 3 (as the driven distance in meters) yields a result of $\frac{1}{3}\sqrt{\frac{1}{4}e_{e_y}} = \frac{1}{6}\sqrt{e_{e_y}} = \overline{|\epsilon_{e_y}|} = 3.11$ millimeter mean absolute value of the errors of the lateral direction of the error mean vector per 1 meter of distance driven backward. This is in addition the largest of the residuals, the other having their values at least 10 times smaller. It yields sub-millimeter and sub-milliradian means of absolute values of the approximation errors of all error mean vector components and values of the mean approximation errors of the covariance matrix parameters of orders of around 10^{-6} of their respective units.

Finally, the approximated contribution to the error mean vector and the vectorized upper diagonal block of the symmetric covariance matrix corresponding to the error mean vector can be expressed as

$$\Delta \begin{bmatrix} e_x & e_y & e_\theta & m_1 & m_2 & m_3 & m_4 & m_5 & m_6 \end{bmatrix}^T (\Delta d, v_x, v_\theta) =$$
$$\begin{bmatrix} \Delta e_x & \Delta e_y & \Delta e_\theta & \Delta m_1 & \Delta m_2 & \Delta m_3 & \Delta m_4 & \Delta m_5 & \Delta m_6 \end{bmatrix}^T \doteq$$

$$\doteq \begin{cases} \frac{1}{d'} \boldsymbol{X}\left(0.5\,\mathrm{m/s}\right) \begin{bmatrix} v_\theta^2 \Delta d \\ v_\theta \Delta d \\ \Delta d \end{bmatrix} & v_x \geq 0 \\ \frac{1}{d'} \boldsymbol{X}\left(-0.5\,\mathrm{m/s}\right) \begin{bmatrix} v_\theta^2 \Delta d \\ v_\theta \Delta d \\ \Delta d \end{bmatrix} & v_x < 0 \end{cases}, \tag{6.35}$$

$$\doteq \begin{cases} \begin{bmatrix} -0.4933\ \frac{\mathrm{s}^2}{\mathrm{rad}^2} & -0.0328\ \frac{\mathrm{s}}{\mathrm{rad}} & 0.0171 \\ -0.0513\ \frac{\mathrm{s}^2}{\mathrm{rad}^2} & -0.2737\ \frac{\mathrm{s}}{\mathrm{rad}} & -0.0038 \\ -0.0366\ \frac{\mathrm{s}^2}{\mathrm{rad\cdot m}} & -0.0104\ \frac{\mathrm{s}}{\mathrm{m}} & -0.0034\ \frac{\mathrm{rad}}{\mathrm{m}} \\ 115.108 \cdot 10^{-6}\ \frac{\mathrm{m\cdot s}^2}{\mathrm{rad}^2} & 20.750 \cdot 10^{-6}\ \frac{\mathrm{m\cdot s}}{\mathrm{rad}} & 1.864 \cdot 10^{-6}\ \mathrm{m} \\ 59.009 \cdot 10^{-6}\ \frac{\mathrm{m\cdot s}^2}{\mathrm{rad}^2} & 11.207 \cdot 10^{-6}\ \frac{\mathrm{m\cdot s}}{\mathrm{rad}} & -0.289 \cdot 10^{-6}\ \mathrm{m} \\ 65.841 \cdot 10^{-6}\ \frac{\mathrm{s}^2}{\mathrm{rad}} & 6.964 \cdot 10^{-6}\ \mathrm{s} & -0.708 \cdot 10^{-6}\ \mathrm{rad} \\ -57.755 \cdot 10^{-6}\ \frac{\mathrm{m\cdot s}^2}{\mathrm{rad}^2} & 11.182 \cdot 10^{-6}\ \frac{\mathrm{m\cdot s}}{\mathrm{rad}} & 9.625 \cdot 10^{-6}\ \mathrm{m} \\ -57.171 \cdot 10^{-6}\ \frac{\mathrm{s}^2}{\mathrm{rad}} & 2.873 \cdot 10^{-6}\ \mathrm{s} & 4.760 \cdot 10^{-6}\ \mathrm{rad} \\ -29.991 \cdot 10^{-6}\ \frac{\mathrm{s}^2}{\mathrm{m}} & 1.283 \cdot 10^{-6}\ \frac{\mathrm{rad\cdot s}}{\mathrm{m}} & 9.996 \cdot 10^{-6}\ \frac{\mathrm{rad}^2}{\mathrm{m}} \end{bmatrix} \begin{bmatrix} v_\theta^2 \Delta d \\ v_\theta \Delta d \\ \Delta d \end{bmatrix} & v_x \geq 0 \\ \begin{bmatrix} -1.2141\ \frac{\mathrm{s}^2}{\mathrm{rad}^2} & 0.0161\ \frac{\mathrm{s}}{\mathrm{rad}} & -0.0155 \\ 0.0012\ \frac{\mathrm{s}^2}{\mathrm{rad}^2} & -0.2551\ \frac{\mathrm{s}}{\mathrm{rad}} & -0.0055 \\ -0.0272\ \frac{\mathrm{s}^2}{\mathrm{rad\cdot m}} & 0.0051\ \frac{\mathrm{s}}{\mathrm{m}} & -0.0032\ \frac{\mathrm{rad}}{\mathrm{m}} \\ 61.347 \cdot 10^{-6}\ \frac{\mathrm{m\cdot s}^2}{\mathrm{rad}^2} & -7.653 \cdot 10^{-6}\ \frac{\mathrm{m\cdot s}}{\mathrm{rad}} & 4.324 \cdot 10^{-6}\ \mathrm{m} \\ -7.491 \cdot 10^{-6}\ \frac{\mathrm{m\cdot s}^2}{\mathrm{rad}^2} & 10.849 \cdot 10^{-6}\ \frac{\mathrm{m\cdot s}}{\mathrm{rad}} & -0.755 \cdot 10^{-6}\ \mathrm{m} \\ -104.060 \cdot 10^{-6}\ \frac{\mathrm{s}^2}{\mathrm{rad}} & -4.541 \cdot 10^{-6}\ \mathrm{s} & 2.801 \cdot 10^{-6}\ \mathrm{rad} \\ 104.179 \cdot 10^{-6}\ \frac{\mathrm{m\cdot s}^2}{\mathrm{rad}^2} & -0.632 \cdot 10^{-6}\ \frac{\mathrm{m\cdot s}}{\mathrm{rad}} & 4.375 \cdot 10^{-6}\ \mathrm{m} \\ 4.545 \cdot 10^{-6}\ \frac{\mathrm{s}^2}{\mathrm{rad}} & -2.389 \cdot 10^{-6}\ \mathrm{s} & -3.025 \cdot 10^{-6}\ \mathrm{rad} \\ 25.804 \cdot 10^{-6}\ \frac{\mathrm{s}^2}{\mathrm{m}} & 1.245 \cdot 10^{-6}\ \frac{\mathrm{rad\cdot s}}{\mathrm{m}} & 8.043 \cdot 10^{-6}\ \frac{\mathrm{rad}^2}{\mathrm{m}} \end{bmatrix} \begin{bmatrix} v_\theta^2 \Delta d \\ v_\theta \Delta d \\ \Delta d \end{bmatrix} & v_x < 0 \end{cases} \tag{6.36}$$

where v_θ is the commanded angular velocity, v_x is the commanded forward velocity, $\boldsymbol{X}\left(v_x\right)$ is the matrix of coefficient expressed by (4.66) depending on the commanded v_x and Δd is the amount of driven distance between 2 subsequent updates of the pose of the EKF-fused odometry and $d' = 3$ is the distance driven to obtain the $\boldsymbol{X}\left(v_x\right)$.

It is also apparent that the non-zero values of the components of the error vector can be used in order to adjust the actual considered pose of the filtered odometry in the SLAM algorithm.

Chapter 7

Suggestions for the Further Use of the Results

The result of the last experiment may be now adapted in the actual SLAM algorithm, which for now does not consider the uncertainty of the pose of the robot estimated by the EKF-fused odometry. Here is a suggestion how to do it.

First, it is needed to determine by comparison of the data of the raw encoder odometry and of the EKF-fused odometry, whether the IMU has been properly initialized or not. If the IMU is initialized properly, after having driven several centimeters from the pose where the robot had been booted up, the positions estimated by the raw wheel odometry and by the EKF-fused odometry should be distant at least 0.1 mm from each other. In other case it means that the IMU has not been properly initialized and therefore the following algorithm should not be used, yet the actual algorithm not considering the uncertainty should be used instead.

If the IMU has been initialized properly, the following algorithm should be used. Assume having remembered the last estimation of the pose denoted by $\boldsymbol{p}_k = \begin{bmatrix} x_k & y_k & \theta_k \end{bmatrix}^T$ and the corresponding last estimated covariance matrix $\boldsymbol{\Sigma}_k$, which is initially a 3×3 matrix of zeros, and having received an update of the estimated pose $\boldsymbol{p}_{k+1} = \begin{bmatrix} x_{k+1} & y_{k+1} & \theta_{k+1} \end{bmatrix}^T$. First, the contribution to the error vector and the covatiance matrix of the estimated pose of the robot $\begin{bmatrix} \Delta e_x & \Delta e_y & \Delta e_\theta & \Delta m_1 & \Delta m_2 & \Delta m_3 & \Delta m_4 & \Delta m_5 & \Delta m_6 \end{bmatrix}^T$ should be computed using the equation **??** from the actually read commanded linear and angular velocities v_x and v_θ, which have to be obtained from the topic `/cmd_vel`, and from the distance Δd driven from the last remembered pose $\boldsymbol{p}_k$ to the newly received estimated pose update $\boldsymbol{p}_{k+1}$. The driven distance can be computed as

$$\Delta d = \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}\,. \tag{7.1}$$

Now, the update of the estimated pose $\boldsymbol{p}_{k+1}$ obtained from the EKF-fused odometry, should be corrected by the estimated error vector as

$$\boldsymbol{p}^*_{k+1} = \begin{bmatrix} x_{k+1} - \Delta e_x \\ y_{k+1} - \Delta e_y \\ \theta_{k+1} - \Delta e_\theta \end{bmatrix} . \tag{7.2}$$

This corrected pose will be used after the next update cycle of the EKF-fused odometry as $\boldsymbol{p}_k$, or as an input to the SLAM algorithm. Also the contribution of the covariance matrix can be used, as by subtracting the error vector, we can consider an unbiased normal distribution of the uncertainty of the estimated pose, represented by the covariance matrix. Then the new covariance matrix, which will be used after the next update cycle of the EKF-fused odometry as $\boldsymbol{\Sigma}_k$, or as an uncertainty input during the following update of the SLAM algorithm, can be computed as

$$\boldsymbol{\Sigma}_{k+1} = \boldsymbol{\Sigma}_k + \begin{bmatrix} \Delta m_1 & \Delta m_2 & \Delta m_3 \\ \Delta m_2 & \Delta m_4 & \Delta m_5 \\ \Delta m_3 & \Delta m_5 & \Delta m_6 \end{bmatrix} . \tag{7.3}$$

Finally, when the update of the SLAM algorithm is about to happen, the gradually corrected pose $\boldsymbol{p}^*_n$ and the corrected covariance matrix $\boldsymbol{\Sigma}_n$ shall be used as the inputs to the update cycle of the slam algorithm, where n denotes the count of updates of the pose estimated by the EKF-fused odometry from the last update of the SLAM algorithm. Finally the SLAM algorithm will return the newly estimated pose, which will be assumed as the new originating pose $\boldsymbol{p}_0$. Also, the new originating covariance matrix $\boldsymbol{\Sigma}_0$ will be set to be a 3×3 matrix of zeros.

Chapter 8

Conclusion

The experiments have revealed some interesting pieces of information about the odometry of the robot and about its behaviour, as well as about the accuracy of the HTC Vive tracking system.

The first experiment has shown that the EKF-fused odometry of the robot composes its estimation of the pose of the robot separately from the data of the wheel odometry and of the IMU, in terms of the distances and orientation angles. It has been observed, that the information about the orientation of the robot is taken purely from the data of the IMU, whereas the information about the driven distance is taken purely from the encoder odometry.

The second experiment has shown that the HTC Vive tracking system is not suitable to be used as a source of the odometry. Besides the already known fact that it is sensible to external light radiation, the accuracy of the tracking system decreases significantly with the increasing distance from the pose, where it has been calibrated.

The third and at the same time the largest experiment has revealed some interesting information about the behavior of the driving of the robot and of the odometries. It has been observed that the robot keeps always subtly steering more left in the direction of its driving, than is commanded, even if it is commanded to drive straight and not to steer at all. It has been also observed that the IMU sometimes does not initialize properly, which seems to be a non-deterministic phenomenon. Yet it has been learned that it can be detected by comparison of the data of the EKF-fused odometry and of the raw encoder odometry. If the data of these 2 odometries are identical even after a few centimeters of driving and steering, regardless of some sub-millimeter floating point errors, then it means that the IMU has not been properly initialized, and therefore the EKF-fused odometry takes only the data of the raw wheel odometry, wherefore the data are identical. It has been therefore observed, that depending on whether the IMU has been properly initialized or not, the error and the uncertainty of the EKF-fused odometry forms 2 distinct probabilistic distributions. The wrong initialization of the IMU can be solved simply by rebooting the robot. Another interesting information, which has been observed and which is caused by the observed slight omnipresent steering, is the fact, that the uncertainty of the EKF-fused odometry during forward driving of the robot is not analogically comparable to the uncertainty when driving backward.

As the result of the third experiment, based on the observations mentioned above, the contribution to the error vector and to the covariance matrix between 2 subsequent updates of the SLAM algorithm, running on the robot, has been approximated. This approximation is a complex dependence, consisting of a binary dependence on the sign of the commanded linear velocity, of a quadratic dependence on the value of the commanded angular velocity and of a pure unbiased linear dependence on the distance driven from the last update of the pose from the EKF-fused odometry. Subsequently, it has been suggested how to use this result in the SLAM to correct the pose taken from the EKF-fused odometry, using the contribution to the error vector, and how to use the contribution to the covariance matrix in order to consider the uncertainty of the estimated pose in the actual SLAM algorithm, which now does not consider any uncertainty of the pose represented by its covariance matrix at all.

Bibliography

[1] Hexagon AB. *Leica Absolute Tracker AT403*. Hexagon AB, 2017. Available at `https://www.creativeinfocom.com/pdfs/leica-absolute-tracker-at403-brochure.pdf`.

[2] Leica Geosystems AG. *Laser Tracker Accessories in All Dimensions*. Leica Geosystems AG, 2002. Available at `https://w3.leica-geosystems.com/media/new/product_solution/L3_Accessories_Laser_Tracke.pdf`.

[3] N. Akai, L. Y. Morales, T. Yamaguchi, E. Takeuchi, Y. Yoshihara, H. Okuda, T. Suzuki, and Y. Ninomiya. Autonomous driving based on accurate localization using multilayer lidar and dead reckoning. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, Oct 2017.

[4] Matěj Boxan. *NDT SLAM Respecting Visibility*. Bachelor thesis, Faculty of Electrical Engineering, Czech Technical University in Prague, May 2020.

[5] Byoung-Suk Choi, Joon-Woo Lee, and Ju-Jang Lee. Localization and map-building of mobile robot based on rfid sensor fusion system. In *2008 6th IEEE International Conference on Industrial Informatics*, pages 412–417, July 2008.

[6] J. Będkowski, M. Pełka, K. Majek, T. Fitri, and J. Naruniec. Open source robotic 3d mapping framework with ros — robot operating system, pcl — point cloud library and cloud compare. In *2015 International Conference on Electrical Engineering and Informatics (ICEEI)*, pages 644–649, Aug 2015.

[7] J. Cai and X. Zhong. An adaptive square root cubature kalman filter based slam algorithm for mobile robots. In *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 2215–2219, Aug 2015.

[8] Xueli Cheng, Wanli Liu, Meng Guo, and Zhenhua Zhang. Mobile robot self-localization based on multi-sensor fusion using limited memory kalman filter with exponential fading factor. *journal of Engineering Science and Technology Review*, 11:187–196, 12 2018.

[9] H. M. Cho, H. Jo, S. Lee, and E. Kim. Odometry estimation via cnn using sparse lidar data. In *2019 16th International Conference on Ubiquitous Robots (UR)*, pages 124–127, June 2019.

[10] H. F. Durrant-Whyte and J. J. Leonard. Navigation by correlating geometric sensor data. In *Proceedings. IEEE/RSJ International Workshop on Intelligent Robots and Systems '. (IROS '89) 'The Autonomous Mobile Robots and Its Applications*, pages 440–447, Sep. 1989.

[11] Z. Fan, C. Li, Y. Wang, L. Zhao, L. Yao, G. Zhu, Z. Li, H. Xie, and Y. Xiao. 3d mapping of multi-floor buildings based on sensor fusion. In *2017 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*, pages 10–15, Dec 2017.

[12] S. Farzad Bahreinian, M. Palhang, and M. R. Taban. Investigation of rmf-slam and amf-slam in closed loop and open loop paths. In *2016 2nd International Conference of Signal Processing and Intelligent Systems (ICSPIS)*, pages 1–5, Dec 2016.

[13] Nghia Ho. Finding optimal rotation and translation between corresponding 3D points, 2011. Available at `http://nghiaho.com/?page_id=671`.

[14] T. T. Hoang, P. M. Duong, N. T. T. Van, D. A. Viet, and T. Q. Vinh. Multi-sensor perceptual system for mobile robot and sensor fusion-based localization. In *2012 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 259–264, Nov 2012.

[15] Daisuke Inoue, Daisuke Murai, Yasuhiro Ikuta, and Hiroaki Yoshida. Distributed range-based localization for swarm robot systems using sensor-fusion technique. In *SENSORNETS*, 2019.

[16] Lukáš Jelínek. *Graph-based SLAM on Normal Distributions Transform Occupancy Map*. Bachelor thesis, Faculty of Mathematics and Physics, Charles University in Prague, September 2016.

[17] S. J. Julier. An empirical study into the use of chernoff information for robust, distributed fusion of gaussian mixture models. In *2006 9th International Conference on Information Fusion*, pages 1–8, July 2006.

[18] K. Komoriya, E. Oyama, and K. Tani. Planning of landmark measurement for the navigation a mobile robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1476–1481, July 1992.

[19] Krzysztof Kozłowski and Dariusz Pazderski. Modeling and control of a 4-wheel skid-steering mobile robot. *International Journal of Applied Mathematics and Computer Science*, 14, 01 2004.

[20] J. Laconte, S. Deschênes, M. Labussière, and F. Pomerleau. Lidar measurement bias estimation via return waveform modelling in a context of 3d mapping. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8100–8106, May 2019.

[21] S. Lee, J. Jung, S. Kim, I. Kim, and H. Myung. Dv-slam (dual-sensor-based vector-field slam) and observability analysis. *IEEE Transactions on Industrial Electronics*, 62(2):1101–1112, Feb 2015.

[22] L. Li, M. Yang, C. Wang, and B. Wang. Rigid point set registration based on cubature kalman filter and its application in intelligent vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 19(6):1754–1765, June 2018.

[23] J. Lin, Y. Liao, Y. Wang, Z. Chen, and B. Liang. A hybrid positioning method for multi-robot simultaneous location and mapping. In *2018 37th Chinese Control Conference (CCC)*, pages 4739–4743, July 2018.

[24] Y. Liu, L. Zuo, C. Zhang, and F. Liu. Fast and accurate robot localization through multi-layer pose correction. In *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 2487–2492, Aug 2019.

[25] Bence Magyar. *diff_drive_controller*. ROS Wiki, 2020. Available at `http://wiki.ros.org/diff_drive_controller`.

[26] Wim Meeussen. *robot_pose_ekf*. ROS Wiki, 2020. Available at `http://wiki.ros.org/robot_pose_ekf`.

[27] David Nováček. *Localization of Mobile Robot Using Multiple Sensors*. Master thesis, Faculty of Electrical Engineering, Czech Technical University in Prague, May 2019.

[28] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot. Conservative to confident: Treating uncertainty robustly within learning-based control. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 421–427, May 2015.

[29] S. Rabiee and J. Biswas. A friction-based kinematic model for skid-steer wheeled mobile robots. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8563–8569, May 2019.

[30] Ian Reid. *Estimation I*. Department of Engineering Science, University of Oxford, Hilary term 2001. Available at `http://www.robots.ox.ac.uk/~ian/Teaching/Estimation/LectureNotes1.pdf`.

[31] Ian Reid. *Estimation II*. Department of Engineering Science, University of Oxford, Hilary term 2001. Available at `http://www.robots.ox.ac.uk/~ian/Teaching/Estimation/LectureNotes2.pdf`.

[32] I. Toroslu and M. Doğan. Effective sensor fusion of a mobile robot for slam implementation. In *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, pages 76–81, April 2018.

[33] C. Uyulan, T. Erguzel, and E. Arslan. Mobile robot localization via sensor fusion algorithms. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 955–960, Sep. 2017.

[34] Tianmiao Wang, Yao Wu, Jianhong Liang, Chenhao Han, Jiao Chen, and Qiteng Zhao. Analysis and experimental kinematics of a skid-steering wheeled robot based on a laser scanner sensor. *Sensors*, 15:9681–9702, 05 2015.

[35] Y. Watanabe and S. Yuta. Position estimation of mobile robots with internal and external sensors using uncertainty evolution technique. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 2011–2016 vol.3, May 1990.

[36] N. Yu and B. Zhang. An improved hector slam algorithm based on information fusion for mobile robot. In *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pages 279–284, Nov 2018.

[37] W. Yuan, Z. Li, and C. Su. Rgb-d sensor-based visual slam for localization and navigation of indoor mobile robot. In *2016 International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 82–87, Aug 2016.

[38] Jing-Shan Zhao, Zhi-Jing Liu, and Jian Dai. Design of an ackermann type steering mechanism. *Journal of Mechanical Engineering Science*, 227, 11 2013.

[39] B. Zhou, Z. Tang, K. Qian, F. Fang, and X. Ma. A lidar odometry for outdoor mobile robots using ndt based scan matching in gps-denied environments. In *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 1230–1235, July 2017.

[40] David Štych. HTC Vive testing. report, Robotic and Machine Perception group, Czech Institute of Informatics, Robotics and Cybernetics, Feb 2020. Available internally at `https://gitlab.ciirc.cvut.cz/stychdav/htc_vive_testing`.

Appendix A

Structure of the CD

```
├─ experiments
│  ├─ 2nd_experiment_data
│  │  └─ VIVE_Leica_measured_calibration_points-1582756196459412097.npy
│  ├─ 3rd_experiment_extracted_data
│  │  └─ *.data.npy - numpy-packed extracted data of the 3rd experiment
│  └─ 3rd_experiment_figures
│     └─ ...  folders containing all the resulting figures from the 3rd experiment
└─ scripts
   ├─ find_transform.py - script implementing the theory from 4.4
   ├─ kbhit.py - external class for reading the keyboard queue mentioned in 5.2
   ├─ odometry_classes.py - odometry classes for use in the scripts
   ├─ plot_path.py - visualization and evaluation script descrined in 5.4
   └─ print_odom.py - measurement and driving script described in 4.3
```

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Plavec Václav**	Personal ID number:	**474436**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Cybernetics and Robotics**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Sensor Fusion for Mobile Robot Localization

Bachelor’s thesis title in Czech:

Slučování informací z více senzorů

Guidelines:

1. Get familiar with the mobile robot Jackal and the Robot Operating System.
2. Search the literature on combining information from different sources.
3. Design and implement algorithm for sensor fusion.
4. Perform experiments and evaluate them.
5. Make conclusions

Bibliography / sources:

1. Ehab I. Al Khatib ; Mohammad A. Jaradat ; Mamoun Abdel-Hafez ; Milad Roigari: Multiple sensor fusion for mobile robot localization and navigation using the Extended Kalman Filter, January 2016, 2015 10th International Symposium on Mechatronics and its Applications
2. Çağlar UyulanÇağlar UyulanTurker Tekin ErguzelTurker Tekin ErguzelErsen ArslanErsen Arslan: Mobile Robot Localization Via Sensor Fusion Algorithms, Intelligent Systems Conference 2017
3. Leonardo Marín *, Marina Vallés, Ángel Soriano, Ángel Valera and Pedro Albertos: Multi Sensor Fusion Framework for Indoor-Outdoor Localization of Limited Resource Mobile Robots, Sensors 2013

Name and workplace of bachelor’s thesis supervisor:

Ing. Vladimír Smutný, Ph.D., Robotic Perception, CIIRC

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **23.09.2019** Deadline for bachelor thesis submission: **06.01.2021**

Assignment valid until: **30.09.2021**

Ing. Vladimír Smutný, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt | Student’s signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC