DTU Elektro
Department of Electrical Engineering

Resilient Farmland Navigation through Sensor Fusion and Condition Monitoring

Master Thesis

Resilient Farmland Navigation through Sensor Fusion and Condition Monitoring

Master Thesis
July, 2022

By
Panagiotis Grigoriadis

Copyright: Reproduction of this publication in whole or in part must include the customary bibliographic citation, including author attribution, report title, etc.
Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Electrical Engineering, Ørsteds Plads, Building 348, 2800 Kgs. Lyngby Denmark
https://www.elektro.dtu.dk/
ISSN: [0000-0000] (electronic version)
ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)
ISBN: [000-00-0000-000-0] (printed version)

Approval

This thesis has been prepared over seven months at the Department of Electrical Engineering: Autonomous Systems, at the Technical University of Denmark, DTU, in partial fulfilment for the degree Master of Science in Engineering, MSc Eng.

It is assumed that the reader has basic knowledge in the area of robotics.

Panagiotis Grigoriadis - s203269

.....

Signature

.....

Date

Abstract

The steadily increasing world population has significantly increased food demand. Therefore, for UN's goal for zero hunger to be met, innovative agricultural solutions need to arise. This also goes hand in hand with UN's goal for resilient infrastructure. In this context, one solution that will help transform this industrial sector and improve its efficiency and sustainability is autonomous agricultural vehicles.

The issue of ensuring accurate and dependable localization of agricultural vehicles is the focus of this thesis. The standard localization solution for autonomous vehicles is using the GPS. This goal of this thesis is to provide an auxiliary localization method using the radar and the IMU. Using this auxiliary method, this thesis also aims to develop a method for detecting faults in the GPS signal and raise relevant alarms.

Two real datasets (training and validation) that were specifically collected with the vehicle of the collaborating company for the purposes of this thesis were used.

The results of the auxiliary localization method on the training dataset were very promising. The fault detection results were also very favourable.

The results on the validation dataset were very troubling.

Acknowledgements

[Roberto Galeazzi], [Associate Professor], [DTU, Automation and Control]
For his invaluable feedback during the thesis meetings that took place every two weeks, as well as his availability for additional physical meetings, when necessary, to help when rather difficult challenges arose.

[Søren Hansen], [Associate Professor], [DTU, Automation and Control]
For his attendance at the thesis meeting and clever inputs on the performance of the used algorithms.

[Dimitrios Dagdilelis], [PhD student], [DTU, Automation and Control]
For his consistent availability, great suggestions on how to tackle specific problems that he had encountered before, and input on assessing the performance of the developed algorithms.

[Tommy Ertbølle Madsen], [Chief Executive Officer (CEO) & Co-founder], [Agrirobot ApS]
For the great opportunity to write this thesis with his company and help acquiring datasets with Agrirobot's mobile robot.

[Dimosthenis Angelis, Nikolaos Karavasilis, Ilias Kappos, Dimitrios Zerzis], [MSc students], [DTU]
For their emotional help and motivation, as well as the random discussions and intelligent feedback regarding different parts of this thesis project. The completion of this project truly wouldn't be possible without them.

Finally and most importantly, I would like to thank my grandparents, parents and sister, who have stood by my side through good and bad times all these years. Their sacrifices paved the way for me to have a blessed life. I truly am forever proud of the people they are, and grateful for the values that they have passed on to me.

Abbreviations

CDF Cumulative distribution function
DOF Degrees of freedom
EKF Extended kalman filter
ESEKF Error state extended kalman filter
FMCW Frequency modulated continuous wave
GNSS Global navigation satelite system
GPS Global positioning system
IID Independent and identically distributed
IMU Inertial measurement unit
KF Kalman filter
LIDAR Light detection and ranging
MLE Maximum likelihood estimation
PDF Probability density function
PFA Probability of false alarm
RO Radar odometry
RANSAC Random sample consensus
RMSE Root mean square error
SSR Sum of squared residuals
VO Visual odometry
WO Wheel odometry

List of Figures

3.1 Radar returns from a single scan and their relationship with the sensor velocity profile. 7
3.2 Extracting the vehicle ego-motion from the sensor velocity profile, using a single-track model with the Ackermann condition. 10

5.1 The three introduced GPS faults that the GLRT attempts to detect. 18

6.1 The IMU does not present accumulating drift. 19
6.2 Radar specifications. 20
6.3 Ground truth position of the vehicle in the map frame during the training dataset. 21
6.4 Big moving targets occluding the radar field of view. 21
6.5 Ground truth position of the vehicle in the map frame during the validation dataset. The vehicle starts at position [0, -0.755] because the map frame origin lies on the frame of the left GPS antenna, which has an offset of [0, 0.755] from the ”base_link” frame. 22
6.6 The rather problematic environment of the validation dataset can be observed in the above four subfigures. 23

7.1 The structure and functionality of the developed ROS2 packages in the form of an rqt graph. 25

8.1 Validating the radar ego-motion performance on three different radar scans (200, 770, 1108). 26
8.2 The sum of squared residuals from the least squares approach during the RANSAC and LSQ stages of the radar ego-motion algorithm. RANSAC detects the inliers, meaning the stationary targets, which enables LSQ to perform a second least-squares approximation, this time with much lower SSR values. 27
8.3 Radar odometry does a really good job at estimating heading velocity. However, when the vehicle hits the break (at the 85-second mark), some relatively large residuals can be observed. 28
8.4 Radar odometry seemingly does a questionable job at estimating yaw rate. Especially when focusing on the residuals, one realizes that the results are not very accurate. 28
8.5 Radar odometry significantly underestimates the right turn that the vehicle takes (by roughly $20°$). This corresponds to an orientation root mean square error (RMSE) of 34.4637. 29
8.6 The ESEKF, as expected, perfectly follows the simulated trajectory. 30
8.7 The three different sensor fusion combinations trajectories, along with the ground truth trajectory. 31
8.8 The effect that a loss in the GPS signal has on the ESEKF estimated 2d trajectory. 32
8.9 Student’s t distribution fits the data much better than the normal distribution. 32
8.10 There is a very large correlation between the drawn residual and the ones within one lag. Even further away, auto-correlation is significant. 33

8.11 The introduced errors to the residuals sequence clearly alter not only the mean of p_0, but also the standard deviation σ_0. 34
8.12 The returned GLRT test statistic $g(k)$ values with window of size 20. 35
8.13 The returned GLRT test statistic $g(k)$ values with window of size 40. 36
8.14 The returned GLRT test statistic $g(k)$ values with window of size 100. . . . 36
8.15 Scipy's lognorm PDF best fits the student's t GLRT g values for $w = 40$. . . 38
8.16 The fault detection thresholds emerge from the theoretical CDF of g and depend on the distribution choice, the determined PFA, and the window size w. 38
8.17 Position and trajectory of the vehicle under a "big ramp" error and fault detection with a student's t GLRT. 39
8.18 Position and trajectory of the vehicle under a "small ramp" error and fault detection with a student's t GLRT. 40
8.19 Position and trajectory of the vehicle under a "big ramp" error and fault detection with a student's t GLRT. 41
8.20 Position and trajectory of the vehicle under a "spoofing" error and fault detection with a student's t GLRT. 41

9.1 Heading velocity estimation performance of the radar ego-motion algorithm on the training dataset in ROS2/C++. 42
9.2 The position estimates of all the different ESEKF sensor fusion combinations. 43
9.3 The x position estimates of the ESEKF. The effect that possible GPS losses or GPS faults have on the 2d trajectory estimation is made especially visible in the bottom left and right figures. 43
9.4 The 2d trajectory estimates of the ESEKF. The dead reckoning case (top right) results in a translation RMSE of 1.1460. 44
9.5 Heading velocity estimation performance of the radar ego-motion algorithm on the validation dataset in ROS2/C++. 45
9.6 Orientation estimates of the ESEKF. There is an unexplained lag between the IMU estimate and the RTK antenna estimate. 46
9.7 Position estimates of the ESEKF. Dead reckoning with radar and IMU is unreliable. 47
9.8 The 2d trajectory estimates of the ESEKF. Dead reckoning is unreliable, while adding radar odometry to the standard GPS+IMU fusion only makes the trajectory estimation worse. The RTK orientation measurement significantly changes the trajectory estimation. 48

List of Tables

5.1 Distributions used throughout the literature when applying a GLRT. 17

8.1 The parameters of the designed GLRT. 39

Contents

Preface . . . ii
Abstract . . . iii
Acknowledgements . . . iv

1 Introduction **1**
1.1 Background and motivation . . . 1
1.2 Problem statement . . . 1
1.3 Project objectives . . . 1
1.4 Literature review . . . 2
1.5 Proposed solution . . . 3
1.6 Project contributions . . . 3
1.7 Thesis overview . . . 3

2 Theory **5**
2.1 Frames of reference . . . 5
2.2 Sensors . . . 5
2.3 Statistical distributions and the GLRT . . . 6

3 Radar odometry for vehicle heading velocity estimation **7**
3.1 Linear velocity estimation . . . 7
3.2 Stationary target detection - RANSAC . . . 8
3.3 Sensor velocity profile estimation - LSQ . . . 9
3.4 Vehicle heading velocity and yaw rate - Ackermann . . . 9
3.5 Tuning parameters . . . 10

4 Sensor fusion with an ESEKF **11**
4.1 Update step . . . 11
4.2 Correction step . . . 12
4.3 Injection of error and reset . . . 13
4.4 3 different state vectors . . . 13

5 GPS/radar fault detection using the GLRT **14**
5.1 The GLRT . . . 14
5.2 Threshold determination . . . 16
5.3 GLRT assumptions . . . 17
5.4 Introduced faults . . . 17
5.5 Window size determination . . . 18

6 Experiments **19**
6.1 Hardware . . . 19
6.2 Datasets . . . 20

7 Implementation in ROS2 **24**
7.1 Radar odometry package . . . 24
7.2 Sensor fusion package . . . 24
7.3 Custom messages package . . . 24
7.4 Rqt graph . . . 24
7.5 Gps2map package . . . 25

8 Results **26**
8.1 Radar ego-motion 26
8.2 Sensor fusion 29
8.3 Fault detection 32

9 Validation **42**
9.1 ROS2/C++ 42
9.2 Validation dataset 45

10 Conclusions **49**

Bibliography **51**

1 Introduction

1.1 Background and motivation

As the world population keeps rising, so does the need for more agricultural products. Couple this with the fact that agricultural labor can be very wearing for the human body, and one can readily see how the need for more efficient agricultural tactics arises. Within this context, automation and information technologies have explosively entered the agricultural market, with the expectation of improving the efficiency, quality and sustainability of the whole process. This means that field robots like autonomous tractors and aerial drones will slowly start seeing the fields.

1.2 Problem statement

Before autonomous robots hit the fields, certain safety guarantees have to be met. Most importantly, a high level resiliency of such systems towards malfunctions and operational anomalies that might damage the production of crops and farming equipment, as well as endanger property and human life is required.

Part of this is ensuring that farming vehicles will be able to operate and maintain correct positioning, even in the face of possible GNSS signal failures. These failures could be caused by malfunctions of the GNSS system components or blockage in the signal transmission (atmospheric variations, environment occlusions, man-made spoofing with malign intentions) (Bhatti and Ochieng 2007, Du et al. 2021). Failures of such nature can lead to the vehicle leaving the field and trespassing neighboring fields or public road.

Another critical aspect of this is monitoring the integrity of the GNSS system, along with all the other perception sensors (camera, radar, lidar) used for obstacle detection. Trusting a faulty perception sensor can lead to a number of issues, such as failing to detect living beings that approach the vehicle, creating faulty maps of the environment and faulty ego-motion estimates.

The above can be combated by using the data from perception sensors to perform vehicle ego-motion estimation and comparing the results to each other's, as well as to GNSS-based ego-motion estimates. This thesis explores how this can be done with the use of (a) radar(s), (an) IMU(s) and (a) GPS(s).

1.3 Project objectives

The objectives of this thesis project are the following:

1. Perform and document a literature study of specific topics relevant to the thesis work.
2. Assess the availability of relevant data sets through private (company owned) and public repositories.
3. Design, develop and implement a method for robust motion estimation from consecutive radar scans.
4. Design, develop, and implement a robust sensor fusion method that integrates IMU measurements of linear acceleration and angular velocity with the motion estimates from the radar.
5. Design, develop, and implement a condition monitoring system that exploits the auxiliary position fixing system to assess the integrity of the GNSS position measurement and the integrity of the radar sensor(s) in real time.

6. Test, verify and validate the developed resilient position fixing system on relevant data sets.
7. Formalize the project findings in a technical report that describes the design process, documents the achieved results, and discusses the pros and cons of the proposed solutions.

1.4 Literature review

1.4.1 Radar ego-motion

Estimating a vehicle's ego-motion using a radar sensor throughout the literature is done with the use of either frequency modulated continuous wave (FMCW) scanning (spinning) radars (i.e. [Barnes and Posner 2020]), or FMCW non-scanning radars (i.e. [Kellner et al. 2013]). These two types of radars are often referred to as automotive radars.

Scanning FMCW radars produce dense point clouds, which enable ego-motion calculation techniques that use feature extraction and matching between consecutive scans in a similar way that visual odometry (VO) techniques do, while non-scanning FMCW radars produce very sparse point clouds that are not very well suited for such approaches.

For the purposes of this thesis, only one non-spinning FMCW radar was available, so the literature review was more focused on previous work that also used a single radar of this type.

[Yokoo, Beauregard, and Schneider 2009] first introduced an odometry algorithm based on radar sensors, but the system is not suited for agricultural scenarios, as it depends on always having a stationary obstacle directly in front of the autonomous vehicle, which is not realistic. [Kellner et al. 2013] later proposed a radar odometry (RO) algorithm for calculating the 2 degrees of freedom (DOF) 2-d vehicle ego-motion (i.e. the heading velocity v and yaw rate ω) that needs only one radar sensor to work. This algorithm utilizes RANSAC (Fischler and Bolles 1981) for discarding moving targets from the returned point cloud, a least squares method for calculating the vehicle's heading velocity and the Ackermann model (Norris 1906) for calculating the vehicle's yaw rate. [Kellner et al. 2014] extended their work, adding an additional radar sensor to their experiments and thus, calculating the full 3 DOF velocity profile of the vehicle (i.e. the velocity in the x and y directions of a coordinate frame [v_x, v_y] and yaw rate ω). Consequently, the research on radar ego-motion exploded and a number of new algorithms and setups were proposed (i.e. Barjenbruch et al. 2015; Rapp et al. 2015; Ghabcheloo and Siddiqui 2018; Doer and Trommer 2020; Kramer et al. 2020; Narula, Iannucci, and Humphreys 2020; Monaco and Brennan 2020; Gao et al. 2021; Kung, Wang, and Lin 2021; Ng et al. 2021; Retan, Loshaj, and Heizmann 2021).

1.4.2 Sensor fusion

The fusion between a high frequency sensor (i.e. IMU, radar) and a lower frequency one (i.e. GNSS sensor) is a solved problem throughout the literature. The most simple approach is using a simple kalman filter (KF), as first introduced by [Kalman 1960]. However, the simple KF assumes that the model and measurement processes are linear, which in most cases is not true. The extended kalman filter (EKF) is an extended form of the KF which addresses this issue by linearizing the state and measurement models around the predicted state (Haug 2012). Moreover, the error-state extended kalman filter (ESEKF) is an extension of the EKF, which performs predictions on the error of the system states, rather than the states themselves and then merges the two (Sola 2017). The ESEKF is proven to have a better performance than the EKF, as the error state dynamics are linear, which is one of the conditions for optimal Kalman filtering (Madyastha et al. 2011).

1.4.3 Fault detection

Fault detection using the GLRT for a sequence of independent random variables that follow a normal distribution is covered by [Blanke et al. 2016]. This is used to detect changes in the mean or/and standard deviation of a gaussian sequence. [Willersrud et al. 2015] modified this approach to work for changes in the mean of a sequence of independent variables that are modelled by a student's t distribution.

1.5 Proposed solution

This MSc project proposes that an auxiliary position fixing system is developed. This will be done by leveraging motion information available through a single or multiple radars for automotive applications in combination with IMU measurements of linear accelerations and angular velocities. This information can be fused with an ESEKF. The purpose of this system will be to take over the localization task when the signal of the primary positioning system (GPS) is lost.

Additionally, a condition monitoring system should be developed. This can operate on the residuals between the auxiliary positioning system's ego-motion estimates and ego-motion estimates from the GPS. Faults on either one of the ego-motion sources can be detected with a GLRT. Consecutively, if available, a third ego-motion source can be utilized to build a 3-party voting system. This will be done to assess the integrity of the GPS positioning system and the radar sensor(s) in real time.

1.6 Project contributions

The contributions of this thesis project are the following:

1. A radar ego-motion algorithm by replicating the work of [Kellner et al. 2013] and an evaluation of its results in an agricultural environment.
2. An ESEKF implementation, as detailed by [Sola 2017], for fusing the position estimates of the GPS sensor, the linear acceleration and angular velocity measurements of the IMU, and the heading velocity estimates from the aforementioned radar odometry algorithm.
3. A student's t-based GLRT (Willersrud et al. 2015) on the heading velocity residuals between the GPS ego-motion estimates and the radar ego-motion estimates to detect faults in the GPS/radar system.
4. A ROS2-foxy, C++ implementation of the investigated algorithms.

1.7 Thesis overview

The structure of this thesis project is the following:

1. **Chapter 1: Introduction** Chapter 1 includes a brief introduction to Agriculture 4.0 and autonomous farming vehicles, the problem statement, the project objectives, a review of the investigated literature, the proposed solution and the project contributions.
2. **Chapter 2: Theory and Equipment** Chapter 2 includes the necessary theoretical knowledge behind some algorithms and sensors that were used for the purposes of this project.
3. **Chapter 3: Radar Ego-Motion** Chapter 3 includes details regarding the radar odometry algorithm that was implemented for vehicle ego-motion estimation. The main aspects of it are outlier rejection with RANSAC and velocity estimation with a Least Squares approach. Limitations, tuning parameters and possible filtering mechanisms are also discussed.

4. **Chapter 4: Sensor Fusion with an ESEKF** Chapter 4 details the ESEKF algorithm that was used for fusing the data from the GPS, IMU and Radar systems.
5. **Chapter 5: Fault Detection with a GLRT** Chapter 5 details the process of building a GLRT to detect malfunctions in the GPS/radar signal.
6. **Chapter 6: Experiments** Chapter 6 details the two different setups that were used for collecting datasets with the Agrirobot robot and their results. The former was used in the ”training phase”, meaning testing while developing the 3 aforementioned algorithms (RO, ESEKF, GLRT), while the later was used in the ”validation phase”, meaning validating the effectiveness of the already developed algorithms in a new scenario that differs in a number of ways. The results of the ”training dataset” were very good, while the results of the ”validation dataset” were bad.
7. **Chapter 7: Implementation in ROS2** Chapter 7 describes how the algorithms that were initially implemented in Python were transferred to ROS2 and C++.
8. **Chapter 8: Results** Chapter 8 contains the results from applying the algorithms described in the previous chapters. This allows to draw a clear, collective picture of what was achieved.
9. **Chapter 9: Validation** Chapter 9 validates two things. First of, using the original, ”training dataset” it validates whether the ROS2/C++ node gives the same results as the python prototypes. Secondly, it validates the implemented algorithms by testing them on a new ”validation dataset”.
10. **Chapter 10: Conclusions** Chapter 10 concludes which project objectives were achieved. It also offers educated assumptions as to what went wrong with the ”validation process”. Finally, possible future work is discussed.

2 Theory

This chapter covers some of the basic knowledge required to follow the chapters and findings of this thesis project. This includes information about the concept of reference frames, used sensors and statistical distributions.

2.1 Frames of reference

Reference frames are an essential tool in robotics, and more specifically, autonomous vehicles. They are abstract coordinate systems and they enable keeping track of local and global rotational and translation information about the movement of a vehicle.

A 3d movement of a vehicle that included translation and rotation can be described by the following equation:

$$R_a^b * p_a + t_a^b = p_b \tag{2.1}$$

, where R_a^b describes the 3d rotation between the reference frames a and b and t_a^b describes the 3d translation.

In autonomous vehicles, it is usually the case that every sensor that is attached to the vehicle has its own reference frame. Apart from that, the body frame is the reference frame whose origin is usually attached to the center of the vehicle and the map frame is a frame of reference set in a 3d point somewhere in the world. The global position and rotation of the vehicle can be obtained with regards to that map coordinate system.

2.2 Sensors

Two different vehicles were used for this thesis. Both of them had a list of sensors attached which included camera(s), lidar(s), radar(s), IMU(s) and GPS(s).

Cameras provide a seemingly continuous visual image stream which is very useful for tasks such as object detection, localization, ego-motion estimation and debugging. Lidars use lasers to emit light and extract range information by measuring the time interval between the transmitter and receiver taking action. Lidars produce very dense 3d point clouds and are commonly used for localization and mapping tasks. Radars produce sparse 3d point clouds which contain radar cross section (RCS), range, radial speed, azimuth and elevation angle information about each returned point inside the point cloud and are used for tasks such as ego-motion estimation, localization and mapping and obstacle detection. One of the main advantages of radars is that they are unaffected by bad weather conditions, whereas cameras and lidars are severely affected by them. IMUs are composite devices that have an accelerometer, a gyroscope and possibly a magnetometer integrated and correspondingly measure linear acceleration, angular velocity and orientation. It is important to note that linear acceleration and angular velocity measurements from the IMU are subject to both bias and noise, which is modelled as gaussian, and have to be accounted for. Finally, GPS is one of the GNSSs that provides geographical position (latitude, longitude, altitude) and time information to a GPS receiver. GPS is very robust and accurate but when the signal gets lost due to heavy occlusions, atmospheric disturbances or a man-made signal attack among others, the need for an alternate means of vehicle localization arises.

2.3 Statistical distributions and the GLRT

For the fault detection part of this thesis, a probability density function that adequately fits a variable that represents velocity residuals had to be selected. On top of that, a generalized likelihood ratio test (GLRT) had to be performed and a threshold to be chosen based on the cumulative distribution function of the returned test statistic 'g' of the GLRT.

For any given value in the sample space, the probability density function (PDF) of a continuous random variable indicates the probability for this random variable to take this specific value. As an extend, the cumulative distribution function (CDF) of a variable at a specific point in the sample space shows the probability that the inspected variable will take a value less than, or equal to that specific value. Moreover, the term likelihood, or likelihood function, describes the joint probability of the observed data as a function of the parameters of the chosen statistical model.

The GLRT is a technique that uses the ratio of the likelihood of two competing statistical models in order to examine how good the fit of one with respect to the other is. It is commonly used to detect a change with an unknown magnitude in a sequence of independent variables, which are described by an underlying distribution. In the case investigated by this thesis, the first of the two competing statistical models is the window of residuals that are being examined for possible changes/faults and represents hypothesis number one (H_1), while the second represents the null hypothesis (H_0) and contains the residuals on which an offline statistical analysis was performed.

3 Radar odometry for vehicle heading velocity estimation

In this chapter, the implementation of the approach described by [Kellner et al. 2013] for estimating the 2 DOF 2d vehicle ego-motion will by analyzed, along with the results from its testing within a specific agricultural environment. This approach consists of 3 parts:

1. Applying RANSAC for outlier (non-stationary targets) detection. From now on, this phase of the algorithm will be referred to as "stationary target detection" or "RANSAC" phase.
2. Applying a least squares approach to extract the radar sensor velocity in the x and y direction of its own reference frame. This phase will be referred to as the "LSQ" phase.
3. Using the vehicle extrinsic configuration and the Ackermann steering model to extract the vehicle's velocity profile (heading velocity v and yaw rate ω). This phase will be referred to as the "Ackermann" phase.

3.1 Linear velocity estimation

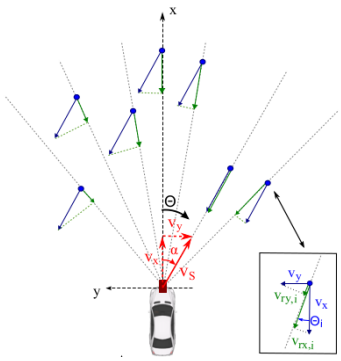


Figure 3.1: Radar returns from a single scan and their relationship with the sensor velocity profile.

Figure 3.1 was taken directly from [Kellner et al. 2013] (fig. 2) and through trigonometry, it becomes evident that for every single returned target i from a single radar scan, the following equation holds:

$$v_{r,i} = \cos(\theta_i) * v_x + \sin(\theta_i) * v_y \tag{3.1}$$

, where for each target i, v_r is its radial speed and θ is its azimuth angle.
This means that if a single radar scan returns N targets, the following linear equation of the form $y = A * x$ will hold:

$$\begin{bmatrix} v_{r,1} \\ \vdots \\ v_{r,N} \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) \\ \vdots & \\ \cos(\theta_N) & \sin(\theta_N) \end{bmatrix} * \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{3.2}$$

, where v_x and v_y are the linear velocities of the radar sensor in the x and y directions of its own reference frame.

Now, the magnitude of the sensor velocity (v_s) and its heading direction (α) can be found by solving the 2x2 system of the following 2 equations:

$$\begin{aligned} v_x &= -\cos(\alpha) v_s \\ v_y &= -\sin(\alpha) v_s \end{aligned} \tag{3.3}$$

, which results in:

$$\begin{aligned} \alpha &= \tan^{-1}(\frac{v_x}{v_y}) \\ v_s &= \frac{-v_x}{\cos(\alpha)} \end{aligned} \tag{3.4}$$

The above procedure is exactly what [Kellner et al. 2013] described and it works fine, because most automotive radars have a very low elevation angle field of view. Specifically, the radar sensor that was used for this project has a $[-7.5°, 7.5°]$ field of view in terms of its elevation angle. For more accurate results, however, and in the presence of larger elevation angle fields of view, one should integrate it into equation 3.1. This equation would then become:

$$v_{r,i} = \cos(\psi_i) * \cos(\theta_i) * v_x + \cos(\psi_i) \sin(\theta_i) * v_y \tag{3.5}$$

, where ψ_i is the elevation angle of radar target i.
The rest of the equations would remain the same.

At this point, it is worth noting that both equation 3.1 and equation 3.5 had indistinguishably good results when testing for the purposes of this thesis project.

The above procedure, or part of it, is used both during the RANSAC and during the LSQ phase of the radar ego-motion algorithm.

3.2 Stationary target detection - RANSAC

Equation 3.2 is valid only for stationary target returns for the radar. This means that a need for a robust method for detecting and rejecting non-stationary targets arises. This can be done using RANSAC, which was first introduced by [Fischler and Bolles 1981]. [Kellner et al. 2013] included this procedure in their novel paper, but for completeness and clearness purposes, it is presented in algorithm 1, in a concise and unambiguous manner.

It is worth noting that at step 5, alternative residuals can be considered. Two such options could be the mean of all the radial speed returns and the magnitude of the sensor velocity profile ($v_x^2 + v_y^2$).

The main disadvantage of this approach is that the described RANSAC algorithm will fail, if the largest group of targets are non-stationary. This means that RANSAC will return a list of inliers that includes moving targets and for which the equation 3.2 no longer applies. This will lead to undefined behavior of the entire radar ego-motion algorithm and an untrustworthy vehicle velocity profile.

The aforementioned scenario is quite rare in agricultural environments, as most of the environment is stationary, but not impossible. The most likely scenario that could lead to this is if a relatively large moving object passed directly in front of the radar sensor, blocking most of its view and leading to most of the radar returns belonging to this very object.

Algorithm 1 RANSAC for radial velocity outlier rejection

```
while True do
    Select either a pair of radar targets returns (case a) or a sub-sample of size S (case b).
    Either solve 3.2 analytically (case a) or using a least squares approximation (case b).
    Calculate the sensor's velocity profile (v_s, α), as described in sections 3.1, 3.2, 3.3.
    Calculate the velocity difference between every v_{r,i} and v_s.
    if Every v_{r,i} is within a predefined tolerance ϵ then
        Exit while loop and return the inlier list.
    else if # of inliers / # of returns is bigger than a predefined ratio r then
        Exit while loop and return the inlier list.
    else if # of iterations is bigger than a predefined number n then
        Exit while loop and return the biggest inlier list so far.
    end if
end while
```

To deal with the above dangerous scenario, Kellner et al. 2013 propose rejecting RANSAC candidates that result in velocity profiles that deviate a lot from the velocity profile of the previous radar scan. This will be referred to as ”velocity filtering” and it is very promising for the agricultural domain, as the top speed of the vehicle is usually known beforehand. Range filtering can also be considered, meaning removing all the radar targets that have a smaller range value return than a predefined threshold. This could be successful because, as mentioned above, objects that are too close to the vehicle are especially concerning because they can block most of the sensor's field of view. Moreover, depending on the agricultural activity, one should know in advance the radius around the vehicle within which nothing should exist, to avoid possible collisions. Finally, similarly to range filtering, RCS filtering could also be considered. This could be done by rejecting radar targets with RCS values larger or smaller than what is usually expected within the agricultural environment that it operates. This could be successful because everything that is detected by the radar which shouldn't normally be there is probably moving in some way. With that being said, RCS filtering was not tested during this project, so its effectiveness is questionable, while range and velocity filtering were tested and produced insignificant results.

In chapter 8, it will be shown that such problematic occurrences can effectively be detected by inspecting the sum of squared residuals (SSR) of the RANSAC fit.

3.3 Sensor velocity profile estimation - LSQ

Equation 3.2 is the core of the second phase of the ego-motion algorithm, as well. After a set of inliers is returned by the RANSAC phase, vector y and matrix A are adjusted accordingly and a least squares approximation is repeated, this time only for the inlier set. The least squares output y is then converted into the velocity profile of the radar sensor (v_s and α) with equation 3.4.

3.4 Vehicle heading velocity and yaw rate - Ackermann

In order to extract the vehicle's heading velocity and yaw rate, one needs to utilize the vehicle extrinsic configuration and Ackermann condition. This is visualized in figure 3.2, which is taken from [Kellner et al. 2013] (fig. 3).

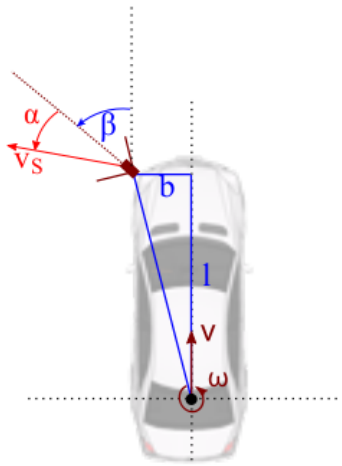


Figure 3.2: Extracting the vehicle ego-motion from the sensor velocity profile, using a single-track model with the Ackermann condition.

In chapter 8, it is shown that in contrast to the heading velocity estimation, the yaw rate estimation isn't very accurate. This could be because, as Kellner et al. 2013 note, "the Ackerman model assumes that there is no wheel drift and thus the velocity of the rear axle contains only an orthogonal part".

3.5 Tuning parameters

Several parameters need to be carefully tuned for the described RO algorithm to work. The most crucial ones are the following:

- The minimum number of radar returns that are required to proceed to the RO algorithm. The smaller this number is, the less dependable RANSAC becomes.
- The maximum number of iterations inside the RANSAC loop. For better results, this should be a large number.
- The residual difference that is allowed between the radial speed of a radar point ($v_{r,i}$) and the sensor's heading velocity (v_s), in order for the point to be considered an inlier. The should be small to better fit equation 3.2, but not too small, so that no inliers are ever found.
- The minimum number of inliers that are required from RANSAC, in order to proceed to the LSQ stage of the RO algorithm. This should also be relatively high for more dependable results.

4 Sensor fusion with an ESEKF

The ESEKF is a well established technique for performing fusion between high frequency and low frequency sensors. As all Kalman Filter approaches, it consists of two steps, the prediction step and the correction/measurement step. In this chapter, these two steps, along with the 3 different types of states (nominal state, error state, true state) used by the ESEKF, will shortly be presented. This thesis part strictly followed the instructions by [Sola 2017] and if the reader seeks a more detailed explanation, they should refer to it.

4.1 Update step

Equations 4.1, 4.2 constitute the update step of the filter.

$$\delta x = -F_x * \delta x \tag{4.1}$$

$$P = F_x * P * {F_x}^T + F_i * Q_i * F_i^T \tag{4.2}$$

, where

- δx is the system error state.
- F_x is the Jacobian of continuous system dynamics with respect to the error, represents the error state kinematics and is called the system transition matrix.
- P is the state covariance matrix.
- F_i is the Jacobian of continuous system dynamics with respect to the perturbation vectors.
- Q_i the covariance matrix of the perturbation impulses.

$$F_x = \begin{bmatrix} I & I * \Delta t & 0 & 0 & 0 & 0 \\ 0 & I & -R * [a_m - a_b]_X * \Delta t & -R * \Delta t & 0 & I * \Delta t \\ 0 & 0 & -R^T\{(\omega_m - \omega_b) * \Delta t\} & 0 & -I * \Delta t & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{bmatrix} \tag{4.3}$$

, where

- R denotes the rotation matrix that corresponds to the current nominal state orientation.
- $R\{\theta\}$ denotes the rotation matrix that corresponds to the euler angle θ.
- $[x]_X$ denotes the skew symmetric matrix that corresponds to the vector x.

$$F_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ I_3 & 0 & 0 & 0 \\ 0 & I_3 & 0 & 0 \\ 0 & 0 & I_3 & 0 \\ 0 & 0 & 0 & I_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.4}$$

$$Q_i = \begin{bmatrix} V_i & 0 & 0 & 0 \\ 0 & \Theta_i & 0 & 0 \\ 0 & 0 & A_i & 0 \\ 0 & 0 & 0 & \Omega_i \end{bmatrix} \tag{4.5}$$

, where V_i, Θi, A_i and Ω_i are the 3x3 covariance matrices that correspond to $v_i, \theta_i, \alpha_i, \omega_i$.

$$V_i = \sigma^2_{\widetilde{\alpha}_n} * \Delta t^2 * I \tag{4.6}$$

$$\Theta_i = \sigma^2_{\widetilde{\omega}_n} * \Delta t^2 * I \tag{4.7}$$

$$A_i = \sigma^2_{\alpha_w} * \Delta t * I \tag{4.8}$$

$$\Omega_i = \sigma^2_{\omega_w} * \Delta t * I \tag{4.9}$$

, where:

- α_n is the noise of the IMU linear acceleration measurement α.
- ω_n is the noise of the IMU angular velocity measurement ω.
- α_w is the noise of the IMU linear acceleration bias a_b.
- ω_w is the noise of the IMU linear acceleration bias ω_b.

Quoting [Sola 2017], "... $v_i, \theta_i, \alpha_i, \omega_i$ are the random impulses applied to the velocity, orientation and bias estimates, modeled by white gaussian processes. Their mean is zero, and their covariance matrices are obtained by integrating the covariances of α_n, ω_n, α_w and ω_w over the step time Δt ... $\sigma_{\widetilde{\alpha}_n}$, $\sigma_{\widetilde{\omega}_n}$, σ_{α_w} and σ_{ω_w} are to be determined from the information in the IMU datasheet, or from experimental measurements."

4.2 Correction step

Equations 4.10, 4.11, 4.12 constitute the correction step of the filter.

$$K = P * H^T * (H * P * H^T + V)^{-1} \tag{4.10}$$

$$\delta x = K * (y - h(x)) \tag{4.11}$$

$$P = (I - K * H) * P \tag{4.12}$$

, where K is the kalman filter gain, H is the jacobian matrix from equation 4.13, V is the covariance matrix of the low frequency measurement sensor, y is the same sensor's measurement and h(x) is the current state estimate (coming from the nominal state) of the corresponding measurement.

$$H = \frac{\partial h}{\partial \delta x} = \frac{\partial h}{\partial \delta x_t} * \frac{\partial x_t}{\partial \delta x} = H_x * X_{\delta x} \tag{4.13}$$

, where

$$X_{\delta x} = \begin{bmatrix} I_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & I_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & Q_{\delta\theta} & 0 & 0 & 0 \\ 0 & 0 & 0 & I_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & I_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_3 \end{bmatrix} \tag{4.14}$$

$$Q_{\delta\theta} = \frac{1}{2} * \begin{bmatrix} -q_x & -q_y & -q_z \\ q_w & -q_z & q_y \\ q_z & q_w & -q_x \\ -q_y & q_x & q_w \end{bmatrix} \tag{4.15}$$

The term H_x depends on the type of measurement. In the presence of a position measurement (i.e. GPS) it becomes 4.16, whereas in when a velocity measurement is available (i.e. velocity estimate from an odometry algorithm) it becomes 4.17.

$$H_{x,pos} = \begin{bmatrix} I_3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.16}$$

$$H_{x,vel} = \begin{bmatrix} 0 & I_3 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.17}$$

The covariance matrix P should be initialized with values that represent the confidence on the values of the initial state of the ESEKF. On top of that, the difference between the values of the Q_i matrix and the values of the P matrix is very significant, as it indicates the relative confidence that the ESEKF has in its kinematics model compared to its correction measurements.

4.3 Injection of error and reset

Immediately after the correction step, the injection of the observed error to the nominal state and the ESEKF reset take place, which are shown in equations 4.18, 4.19, 4.20.

$$x = x_t \tag{4.18}$$

$$\delta x = 0 \tag{4.19}$$

$$P = P \tag{4.20}$$

4.4 3 different state vectors

The ESEKF uses 3 different states (state vectors). The nominal state (eq. 4.21), the error state (eq. 4.22) and the true state (eq. 4.23). Assuming that the ESEKF operates on a vehicle, the nominal state represents the state of the vehicle on a map frame. On the other hand, the error state represents the estimated error that accumulates between correction steps, while only the high frequency sensor(s) operate. The true state integrates error state into the nominal state, right before itself being regarded as the current nominal state. A very important difference is that the nominal and true state use the quaternion form to represent the orientation, while the error state uses the euler representation.

$$x = \begin{bmatrix} p & v & q & a_b & \omega_b & g \end{bmatrix}^T \tag{4.21}$$

$$\delta x = \begin{bmatrix} \delta p & \delta v & \delta q & \delta a_b & \delta & \omega_b & \delta g \end{bmatrix}^T \tag{4.22}$$

$$x_t = \begin{bmatrix} p_t \\ v_t \\ q_t \\ a_{b,t} \\ \omega_{b,t} \\ g_t \end{bmatrix} = x \oplus \delta x = \begin{bmatrix} p + \delta p \\ v + \delta v \\ q * \delta q\{\delta\theta\} \\ a_b + \delta a_b \\ \omega_b + \delta\omega_b \\ g + \delta g \end{bmatrix} \tag{4.23}$$

, where:

- $q\{\theta\}$ stands for the quaternion representation of the euler angle θ.
- $q_1 * q_2$ always stands for quaternion multiplication, as defined in [Sola 2017].

5 GPS/radar fault detection using the GLRT

The GLRT is a commonly used fault detection technique that detects a change in a sequence of random variables with unknown magnitude and direction of change. Specifically for this thesis, a GLRT with an underlying student's t distribution was used to monitor the vehicle's heading velocity residuals. These residuals are created by using the signals from the GPS and the radar. This chapter describes this procedure.

5.1 The GLRT

Since the robot which was used for this project included several different possibilities for extracting a velocity estimate, the GLRT could be used on the velocity residuals between them to detect a fault in one of them. For example, one could imagine the scenario where the robots includes velocity estimates from the GPS, a visual odometry (VO) node and a radar odometry (RO) node. If the GLRT between the RO and GPS estimates showed an error, one could validate which one out of these two sensors is faulty by either performing two more GLRTs between the RO and VO estimates and between the GPS and VO estimates and observing which one out of these two shows an error as well. Another, simpler and less computationally expensive approach could be to just assume that one out of these three sensors (i.e. the camera) is fault-free and after the GLRT between the RO and GPS estimates shows an error, conclude that the faulty sensor is the one whose velocity estimate differs from the VO estimate the most.

That said, because the robot used for this project did not include a VO or a wheel odometry (WO) node, a 3-way voting scheme was impossible. The approach that this thesis followed was to assume that the radar is fault-free and all the errors came from the GPS. This means that the GLRT interprets all errors as GPS errors, which is also easier to model, as one can easily add some noise and/or bias to the GPS data, while it is much trickier to do the same for the radar pointcloud readings.

Specifically, the test statistic that the GLRT was designed upon was the heading velocity residual between the GPS estimate and the RO estimate $v_{res} = v_{gps} - v_{RO}$. This specific residual was chosen for the following reasons:

- The RO algorithm described in chapter 3 and whose results can be seen in chapter 8 returns an accurate heading velocity estimate.
- The other two available sources of measurements regarding the vehicle's 2d translation are the IMU's linear acceleration measurements and the GPS' position measurements. Out of the two, the GPS is far more accurate. Apart from that, this thesis is specifically interested in detecting failures in the GPS signal.
- To compare the GPS' position measurements and the RO's heading velocity estimates, one would either have to differentiate the former and take its euclidean norm or fuse the latter with an orientation measurement and integrate it. Out of these two options, the former is much simpler and does not depend on an external measurement which will, unavoidably, also add noise.
- The term $|v_{gps} - v_{RO}|$ neglects the sign, but there are no guarantees that the residuals will behave equally in the upper and lower tails of their distribution.
- The terms $v_{gps} - v_{RO}$ and $v_{RO} - v_{gps}$ are equivalent.

The null hypothesis H_0 of the test states that no errors were detected in the GPS signal, while the alternative hypothesis H_1 states that an error was present within the examined window of residuals. In particular, this thesis assumes that this error is a change of unknown magnitude and direction in the mean of the sequence of tested residuals. The rest of the underlying distribution parameters of the sequence are assumed unchanged, in the same way that it is done by [Willersrud et al. 2015][1].

$$\begin{aligned} H_0 &: \hat{\theta} \sim t(\mu_0, \sigma, \ldots) \\ H_1 &: \hat{\theta} \sim t(\mu_1, \sigma, \ldots) \end{aligned} \tag{5.1}$$

, where:

- $\hat{\theta}$ is the heading velocity estimate.
- $t(\mu_i, \sigma, \ldots)$ represents the students' t distribution that models the sequence of heading velocity estimates, along with its statistical parameters.

The standard GLRT form is the log-likelihood ratio between the two competing hypotheses H_0 and H_1 (Blanke et al. 2016)[2]:

$$S_j^k(\theta_1) = \sum_{i=j}^{k} \ln \frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i))} \tag{5.2}$$

, where:

- i is the counter corresponding to the investigated residual.
- j is change time.
- k is a given time instant.
- θ_i resembles the parameters of the distribution that fits the data.
- p is the probability density function that fits the data.
- Index $_0$ indicates the null hypothesis and $_1$ indicates the alternative hypothesis

The decision function of the above GLRT, when restricting the fault occurrence time to the last M time instants (M or w will, from here on, refer to the "window size"), produces the test statistic of the GLRT ($g(k)$, or simply g) and takes the following form (Blanke et al. 2016):

$$g(k) = \max_{k-M+1 \leq j \leq k} \max_{\theta_1} S_j^k(\theta_1) \tag{5.3}$$

Equation 5.3 for a sequence of variables that are modelled by a gaussian distribution becomes equation 5.4 (Blanke et al. 2016), while for variables modelled by a student's t distribution it becomes equation 5.5 (Willersrud et al. 2015).

$$g(k) = \frac{1}{2\sigma^2} \max_{k-M+1 \leq j \leq k} \frac{1}{k-j-1} [\sum_{i=j}^{k} z_i - \mu_0)]^2 \tag{5.4}$$

, where:

- σ is the standard deviation of the H_0 sequence (assumed unchanged).
- μ_0 is the sequence mean corresponding to H_0.
- $z(i)$ is the residual observations within the tested window.

[1] The heading velocity residuals are shown to follow a students' t distribution in subsection 8.3.1. Furthermore, in subsection 8.3.2, it is shown that these assumptions do not hold entirely. That said, the GLRT can still be applied under the drawbacks discussed in section 5.3.

[2] Equation 5.2 is valid only under the two main assumptions stated in section 5.3.

$$g(k) = \max_{k-M+1 \leq j \leq k} \frac{p+v}{2} \sum_{i=j}^{k} [-\ln(1 + \frac{1}{v}(\hat{z}(i) - \hat{\mu}_1)^T S^{-1}(\hat{z}(i) - \hat{\mu}_1)) + \ln(1 + \frac{1}{v}(\hat{z}(i) - \hat{\mu}_0)^T S^{-1}(\hat{z}(i) - \mu_0))] \tag{5.5}$$

, where:

- i is the counter corresponding to the investigated residual.
- p is the number of dimensions ($p = 1$ in the uni-variate case).
- v is the degrees of freedom of the fitted t distribution during H_0 (assumed unchanged).
- S is a $p \times p$ matrix and it is a parameter of the multivariate t distribution ($S = 1$ in the uni-variate case).
- $\hat{z}(i)$ is the residual observations within the tested window.
- μ_0 is the sequence mean corresponding to H_0.
- $\hat{\mu}_1$ is the sequence mean corresponding to H_1, which is found by equation 5.6.

$$\hat{\mu}_1 = \frac{1}{k-j+1} \sum_{i=j}^{k} \hat{z}(i) \tag{5.6}$$

The g values that come out of the GLRT indicate how likely it is that the current set of residuals is modelled by the distribution p_0 that corresponds to H_0. Larger values mean that it is very unlikely that the current set belongs to H_0, while smaller values the exact opposite. To create a fault detection alarm, a threshold h needs to be determined, which acts as shown below:

$$\begin{aligned} &g(k) < h : \text{No fault detected} \\ &g(k) >= h : \text{Fault detected at time } k \end{aligned} \tag{5.7}$$

5.2 Threshold determination

If the independent and identically distributed (IID) assumption presented in section 5.3 is met by the sequence of investigated residuals, then the threshold can be found from equation 5.8 (Galeazzi, Blanke, and Poulsen 2012).

$$P_{FA} = \int_{\theta : L_G(\theta) > \gamma} p(\theta; H_0)\, d\theta \tag{5.8}$$

, where:

- P_{FA} (or PFA) is the desired probability of false alarm. This corresponds to how often it is acceptable to raise an alarm indicating that there is something wrong with the GPS sensor, when in fact there is not. It is a tuning parameter and depends on the application.
- $L_G(\theta)$ is the likelihood ratio of the PDFs of the two competing hypotheses $\frac{p(\hat{\theta_1}; \mu_1, H_1)}{p(\theta_0; \mu_0, H_0)}$.

On the other hand, if the IID assumption is not met, then the threshold value can be found from an analysis on the CDF of the distribution of the g values under H_0 (5.9), but the detector will be suboptimal (Galeazzi, Blanke, and Poulsen 2012).

$$P_{FA} = \int_{g : g > h} p(g; H_0)\, dg \tag{5.9}$$

, where:

- g is the GLRT test statistic.
- h is the threshold value which is derived from the CDF of g.

In this context, it is worth pointing that the IID assumption is almost never met in real-life applications. For example, in [Fang, Blanke, and Leira 2015], although a Gaussian GLRT was used, the test statistic was found to follow a Weibull distribution. If the IID had been met, then the GLRT test statistic would have followed a χ^2 distribution (Kay 1998).

Table 5.1 shows different GLRT and the GLRT test statistic distributions that were used by researchers in previous works.

work	GLRT distribution	g distribution
Galeazzi, Blanke, and Poulsen 2012	Double Weibull	LogNormal
Hansen and Blanke 2014	Cauchy	Weibull
Willersrud et al. 2015	Students' t	Weibull
Fang, Blanke, and Leira 2015	Gaussian	Weibull
This thesis	Student's t	Lognormal

Table 5.1: Distributions used throughout the literature when applying a GLRT.

Similarly, because the IID assumption is not met by the heading velocity residuals of this thesis project (subsection 8.3.2), the threshold h determination is also based on the CDF of the distribution of g.

In detail, threshold determination is done by analyzing the test statistic g of the GLRT and the distribution that best fits it. Having found the PDF that best fits the g values, the next step is to determine the PFA and the threshold value h. This can be done by drawing the CDF plot of the chosen distributions and observing for which g value the theoretical CDFs crosses the PFA threshold line. The PFA threshold line is a horizontal line at the probability of value $100 * (1 -$ PFA$)$.

5.3 GLRT assumptions

The GLRT operates under two main assumptions:

1. All the sequence samples are drawn from the same distribution.
2. The sequence samples are independent.

These two assumptions can be combined in one, which is usually stated as: "The samples are independent and identically distributed" (IID assumption).

Apart from the main two assumptions of the GLRT, the implementation of the GLRT that this thesis used assumes that the introduced error to the GPS will not affect any parameter other than the mean of the underlying distribution.

5.4 Introduced faults

Three artificial errors were introduced to the GPS position readings in order to test the effectiveness of the fault detection algorithm. These are the following:

1. "Small ramp" - A ramp noise with magnitude 0.5 and standard deviation of 0.1 applied to the position readings of the GPS across the x axis.

2. ”Big ramp” - A ramp noise with magnitude 1.5 and standard deviation of 0.1 applied to the position readings of the GPS across the x axis.
3. ”Spoofing” - A gradually increasing error whose purpose is to imitate a man-made spoofing action. This is also added to the position reading across the x axis.

The above faults can be seen in figure 5.1.

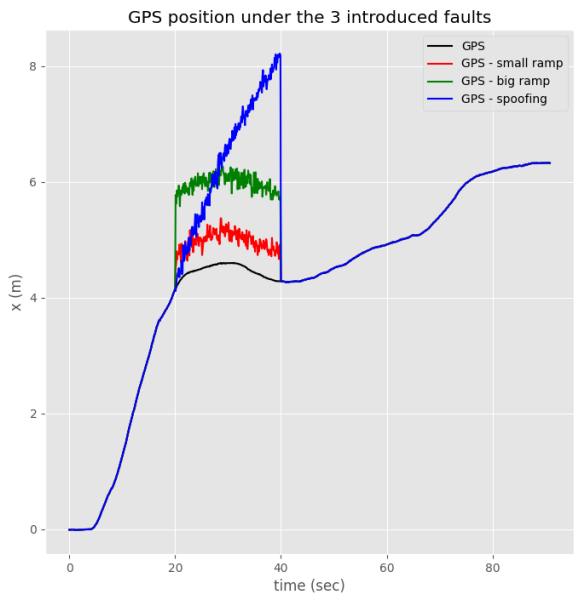


Figure 5.1: The three introduced GPS faults that the GLRT attempts to detect.

The errors are not chosen arbitrarily. Since the positioning margin of error is 2 meters, the minimum GPS fault that should be detected across the 2-dimensional plane is 2 meters. However, since this is a safety application, it is always better to design things with extra caution. Therefore, the ”big ramp” error with a mean of 1.5 corresponds to the minimum error that should be detected. It also leads to the largest possible error detection threshold value. The ”small ramp” error will result in a smaller threshold and lead to detecting even smaller faults in the GPS. Finally, ”spoofing” error gradually leads to a 4 meters error, overtaking the 2-meter margin half way through the sequence window.

This thesis project proposes that the threshold that corresponds to the ”small ramp” error is used by the robot, for additional safety.

5.5 Window size determination

The window size M is a tuneable parameter. When tuning M, three things should be taken into consideration.

1. Smaller window sizes increase the probability of falsely detected faults.
2. Larger window sizes increase the ”detection time” of faults.
3. Larger window sizes add a significant computational overload, which could render the execution of the algorithm in real-time unfeasible.
4. Larger window sizes increase the time it takes for the value of g to return to ”normal” levels that do not indicate an error, once the fault is gone. This will, from now on, be referred to as ”recovery time”.

6 Experiments

This chapter describes the hardware setup that was used on the robot when collecting the datasets that were used for this thesis project. On top of that, the datasets themselves will be shortly presented. The original dataset that was used during development is referred to as ”training dataset”, whereas the one used for validating the results of the training phase is referred to as ”validation dataset”.

6.1 Hardware

In this section, the hardware setup is covered. Sensors that were not utilized for the purposes of this thesis will not be covered, as they bear no significance (i.e. cameras and lidars).

6.1.1 Training

The vehicle with which the training dataset was collected had 1 radar sensor, 1 IMU sensor and 1 GPS sensor mounted. The IMU included an accelerometer and a gyroscope, but no magnetometer.

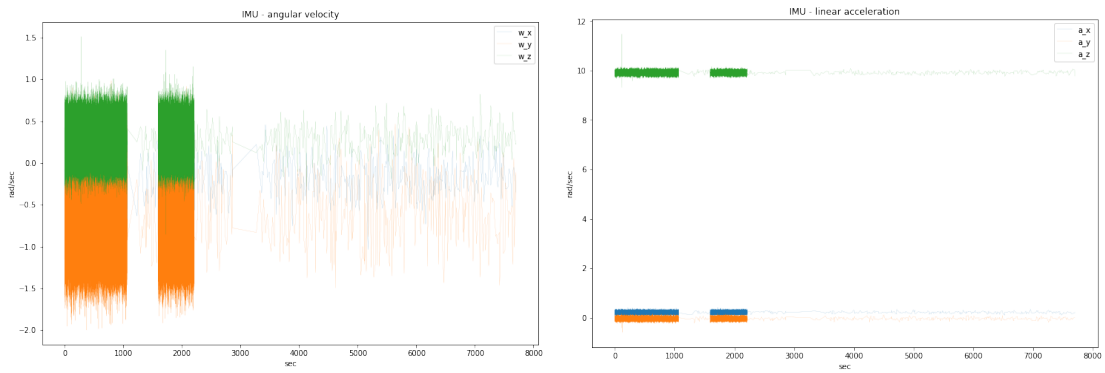
GPS

The GPS antenna that was used by the training vehicle was the Trimble EZ 250.

IMU

The IMU that was used by the training vehicle is the one integrated in the Ouster OS0-128 lidar.

A static test to determine its drift over time was performed over a period of 2 hours. The results can be seen in figure 6.1



(a) IMU angular velocity measurements present no drift over time.

(b) IMU linear acceleration measurements present no drift over time.

Figure 6.1: The IMU does not present accumulating drift.

Two remarks needs to be made regarding figure 6.1:

- IMUs are subject to accumulating drift that needs to be taken into account when using their measurements. If they don’t, this is because of post-processing on their returns. After contacting the company that makes the sensor, it was confirmed that the IMU returns already account for and remove the accumulating drift.

- The IMU readings seem to become sparser after the 1000-second mark. This is obviously an error that fortunately did not have an effect on the conclusions of this test. This error was not encountered again during the collection of datasets and was, therefore, ignored.

Since the IMU presents no accumulating drift, its readings have a stationary bias subject to white noise. The bias for the angular velocity and linear acceleration readings can therefore easily be found by taking the mean of the IMU readings of the above test. This way, it was found to be the following:

$$\begin{aligned} \omega_{bias} &= \begin{bmatrix} -0.00281234 & -0.00941906 & 0.00504196 \end{bmatrix}^T \\ \alpha_{bias} &= \begin{bmatrix} 0.21330168 & -0.04087418 & 9.91409073 \end{bmatrix}^T \end{aligned} \tag{6.1}$$

Radar
The radar that was used by the training vehicle is the UMRR-96 Type 153 provided by Smartmicro. It is a non-spinning FMCW automotive radar.

Its specifications can be seen in figure 6.2. It was configured in medium-range mode.

Parameter		Long-Range Mode	Medium-Range Mode	Short-Range Mode
Operating Frequency		77...81GHz | 3 center frequencies (bands)	77...81GHz | 3 center frequencies (bands)	77...81GHz | 3 center frequencies (bands)
Range	Min./Max.[1]	0.8m/120m | 2.6ft/394ft	0.4m/55m | 1.3ft/180ft	0.15m/19.3m | 0.5ft/63ft
	Separation	< 1.2m | < 3.9ft	< 0.6m | < 2.0ft	< 0.3m | < 1.0ft
	Accuracy	< 0.5m | < 1.64ft or 1% (bigger of)	< 0.3m | < 1.0ft or 1% (bigger of)	< 0.15m | < 0.5ft or 1% (bigger of)
Speed	Min./Max.	-340...+140km/h | -211...+87mph	-340...+140km/h | -211...+87mph	-400...+140km/h | -249...+87mph
	Separation	< 0.3m/s	< 0.3m/s	< 0.3m/s
	Accuracy	< 0.15m/s	< 0.15m/s	< 0.15m/s
Angle	Field of View: Azimuth[2]	-50...+50° (squint beam)	-65...+65° (straight beam)	-65...+65° (straight beam)
	Field of View: Elevation[2]	-7.5...+7.5°		
	Separation: Azimuth	~30° (optional)		
	Accuracy: Azimuth[3]	≤ 1° (at <50° from bore sight)		
	Accuracy: Elevation[3]	≤ 2° (at <10° from bore sight)		

Figure 6.2: Radar specifications.

6.1.2 Validation

The validation dataset vehicle had many more mounted sensors than the training one. These included 5 radar sensors, a dual GPS RTK antenna, instead of a single GPS antenna and 1 IMU. Out of the 5 radar sensors, only one, the front right one, was utilized by this thesis. Moreover, the extra GPS receiver allowed for orientation measurements. Finally, the IMU included an accelerometer and a gyroscope, but no magnetometer.

GPS
The dual GPS RTK system that was used came from the Piksi Multi Evaluation Kit

IMU
The same as in the training dataset.

Radar
The same as in the training dataset.

6.2 Datasets

In this section, an overview of the datasets that were used will be provided. Apart from that, certain characteristics and drawbacks that they present will be discussed.

6.2.1 Training

The dataset used for the training phase was collected in a field and it lasts roughly 90 seconds. The vehicle is surrounded by medium-sized trees. The ground truth position, as obtained from the GPS, can be seen in figure 6.3.

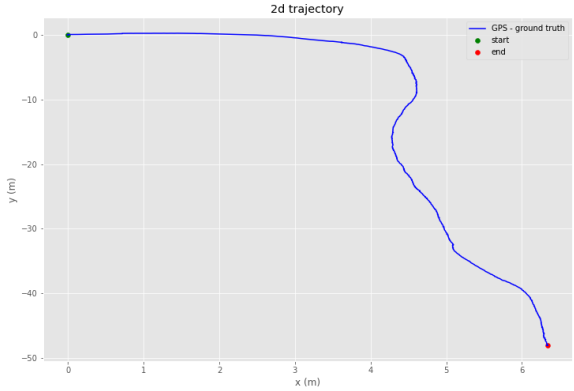


Figure 6.3: Ground truth position of the vehicle in the map frame during the training dataset.

Extra attention should be paid on two different parts of the dataset. Around 40 seconds in and then again at the 60-second mark, people are moving right in front of the vehicle, occluding a bit part of the radar's field of view (figure 6.4). This can have a severe effect on the accuracy of the radar ego-motion algorithm.

(a) Moving targets at the 40-second mark.

(b) Moving targets at the 60-second mark.

Figure 6.4: Big moving targets occluding the radar field of view.

There is one very important drawback regarding the training dataset. This is the fact that it does not include an orientation measurement. This means that the initial orientation measurement of the vehicle needs to be hardcoded and manually tuned. It also means that no orientation corrections can be performed from the ESEKF. This issue, from now on, will be referred to as the "**orientation issue**".

6.2.2 Validation dataset

The dataset used for the validation phase was collected in a parking lot and it lasts roughly 120 seconds. The ground truth position, as obtained from the GPS, can be seen in figure 6.5.

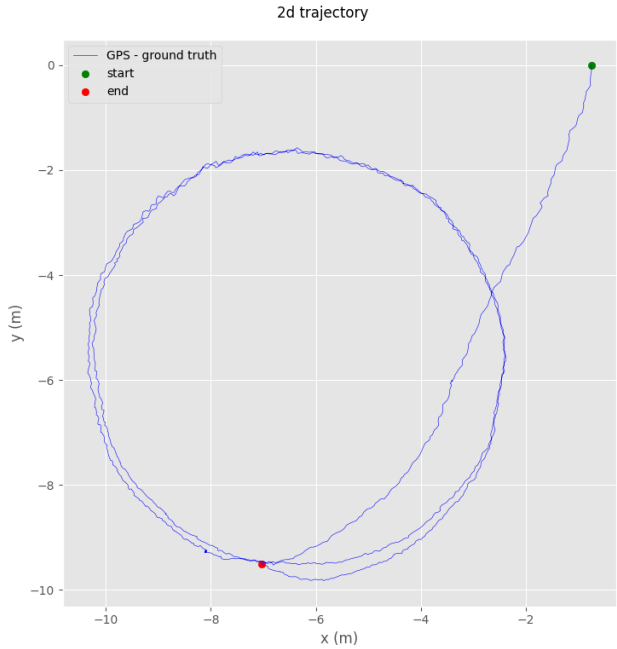


Figure 6.5: Ground truth position of the vehicle in the map frame during the validation dataset. The vehicle starts at position [0, -0.755] because the map frame origin lies on the frame of the left GPS antenna, which has an offset of [0, 0.755] from the ”base_link” frame.

There are three major concerns regarding the validation dataset:

1. It is collected in a completely different environment than the training one (parking lot instead of field).
2. The environment can be too empty, leading to very poor radar pointclouds.
3. Moving targets frequently block the radar’s field of view, which can lead to very unstable radar ego-motion.
4. The vehicle stops moving frequently, which in chapter 8 and during the training dataset is seen to cause overestimation of the vehicles heading velocity by the radar ego-motion algorithm. This can also cause false alarms during the fault detection part, as it will lead to a large number of residuals.

The first three concerns can easily be observed in figure 6.6

(a) Almost empty parking lot environment.

(b) During certain parts of the spiral trajectory of the vehicle, the environment becomes richer, as trees appear in the surroundings.

(c) Moving targets in front of the front right radar sensor (1).

(d) Moving targets in front of the front right radar sensor (2).

Figure 6.6: The rather problematic environment of the validation dataset can be observed in the above four subfigures.

Unfortunately, the author of this thesis had very little control over the type of experiments performed, hence the poor validation dataset quality.

7 Implementation in ROS2

This chapter describes the architectural choices and code structure that was used to transfer all the developed algorithms to C++ and ROS2. It also covers the three different packages that were created.

7.1 Radar odometry package

The ”radar_odom” package includes the radar edo-motion algorithm. Whenever a radar signal is received through a subscriber, the three-part radar-egomotion algorithm takes place. A vehicle velocity profile is then published in the form of the custom message ”RadarOdometryStamped”.

7.2 Sensor fusion package

The ”sensor_fusion” package includes the ESEKF implementation. The GLRT fault detection algorithm is also integrated to it. The main node of this package subcribes to 3 subscribers, one for every sensor that is used by the ESEKF (IMU, radar-RO, GPS). When the IMU subscriber receives a message, the update phase of the kalman filter is triggered, whereas when a RO/GPS message is received, a velocity/position correction step is performed by the kalman filter. The nominal state of the ESEKF is published after every step in the form of the custom message ”EsekfStateStamped”.

It is important to note that if a dual GPS RTK antenna is available (validation dataset), it can be used in two ways. First off, it should be used to initialize the orientation of the vehicle in the map frame. Secondly, it can continue to be used as a correction step for the orientation of the vehicle.

On the other hand, if a vehicle orientation measurement is not available (training dataset), the initial orientation of the vehicle needs to manually be tuned.

7.3 Custom messages package

The ”custom_msgs” package is a helper package that was developed to contain all the different custom ROS2 messages that were used for the purposes of this thesis project.

7.4 Rqt graph

All the above can much more easily be visualized with the ROS tool ”rqt_graph” (figure 7.1).

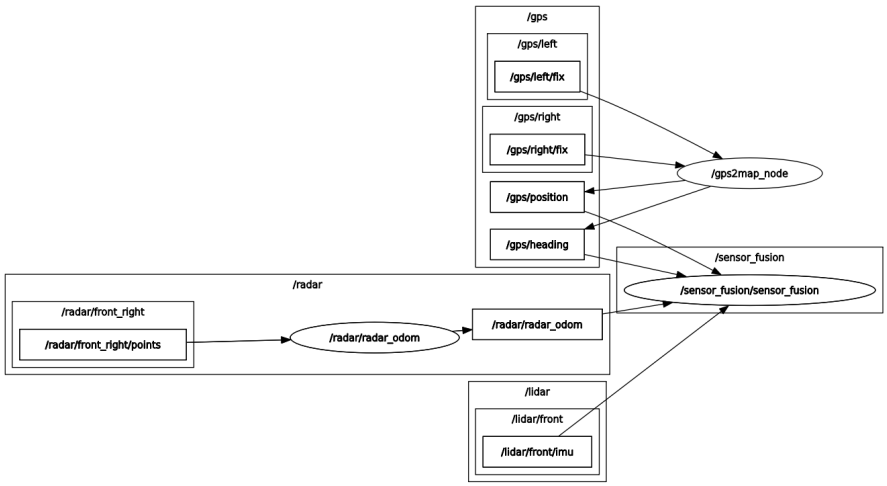


Figure 7.1: The structure and functionality of the developed ROS2 packages in the form of an rqt graph.

7.5 Gps2map package

The ”gps2map” package was not developed by the author of this thesis. It was given to him by the collaborating company and its functionality will be briefly discussed to clear up any ambiguities caused by the rqt graph in figure 7.1.

This package initializes the map frame on the first measurement received by the left GPS antenna. It also subcribes to and synchronizes the messages from the two GPS antennas in order to extract and publish a yaw measurement. Furthermore, it publishes the 2d vehicle ”base_link” position. Both the yaw and position measurements are published with respect to the map frame.

8 Results

This chapter showcases the results from the different algorithms this thesis implemented.

8.1 Radar ego-motion

8.1.1 RANSAC and LSQ - line fit

The RANSAC step of the radar ego-motion algorithm can be validated and evaluated by inspecting the plot showed in figure 8.1. It is the same plot that [Kellner et al. 2013] used in their original paper (figure 4).

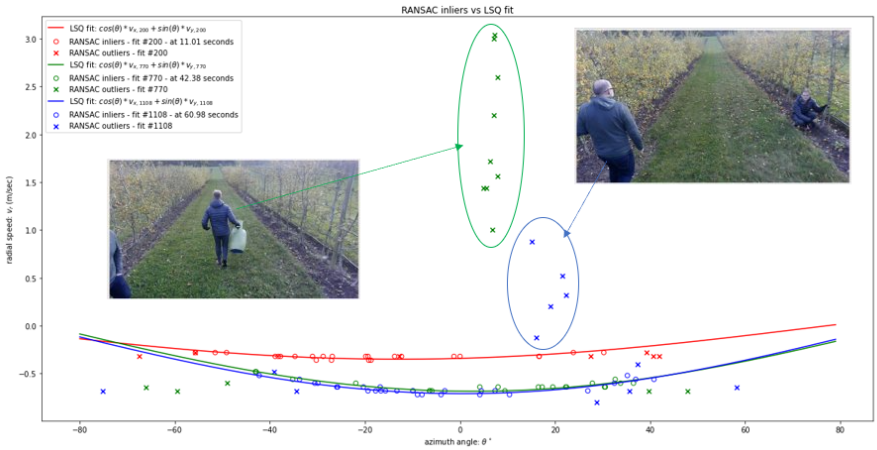


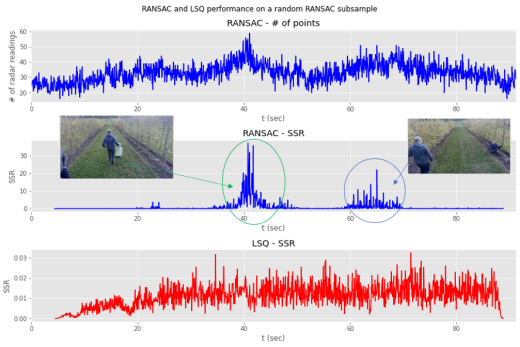
Figure 8.1: Validating the radar ego-motion performance on three different radar scans (200, 770, 1108).

It is a good idea to elaborate on figure 8.1. In this plot, three different radar scans (200, 770, 1108) are inspected. The inliers and outliers for these scans are determined by the RANSAC phase (shown in the figure as 'o' and 'x' scatter symbols). During the LSQ phase, the sensor velocity profile (v_x, v_y) is extracted. Following that, equation 3.1 can be utilized, along with the LSQ velocity profile and a long enough range of azimuth angles (i.e. [-80, 80]) to plot a line (shown in the figure as a continuous line). This line should almost perfectly fit the RANSAC inliers, whilst the outliers should lie away from it.

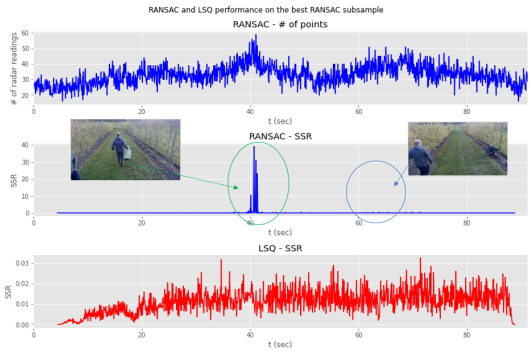
Given a stationary environment, all radar returns should lie very close to the LSQ line fit, which is what happens at fit 200. However, during fit 770 and 1108 one can observe a large amount of outliers, which is caused by moving targets in the range of view of the vehicle.

8.1.2 RANSAC and LSQ - sum of squared residuals

The performance of the RANSAC and LSQ stages of the radar ego-motion algorithm can also be validated by inspecting the SSR returns of the least squares approach on equation 3.2. This can be seen in figure 8.2.



(a) For a random RANSAC sub-sample, the moving targets are detected by the spikes in SSR values around the 40 and 60-second marks. At around 25 seconds, a small spike can be observed probably because of noise in the radar returns, since no moving objects were found in the radar field of view.



(b) For the best RANSAC sub-sample (best is the one that leads to the largest number of inliers), the spikes at the20 and 60 second mark have disappeared, while the one at the 40-second mark persists.

Figure 8.2: The sum of squared residuals from the least squares approach during the RANSAC and LSQ stages of the radar ego-motion algorithm. RANSAC detects the inliers, meaning the stationary targets, which enables LSQ to perform a second least-squares approximation, this time with much lower SSR values.

The fact that in figure 8.2b, the spike at the 60-seconds mark disappeared is a little worrying. It means that a line fit in the form of 3.1 was found that leads to low SSR values even for the radar returns that correspond to moving targets. However, this could be exaplained by the fact that the moving target at 60 seconds is very slowing moving forward, while the target at 40 seconds is running. This is also confirmed to an extend by the plot in figure 8.1.

8.1.3 Vehicle ego-motion

The performance of the algorithm in terms of the validity of its heading velocity and yaw rate estimates can be assessed by comparing them to the GPS estimates and IMU measurements respectively. The GPS heading velocity is taken by differentiating GPS position measurements with respect to time and taking their norm. This can be seen in figures 8.3 and 8.4

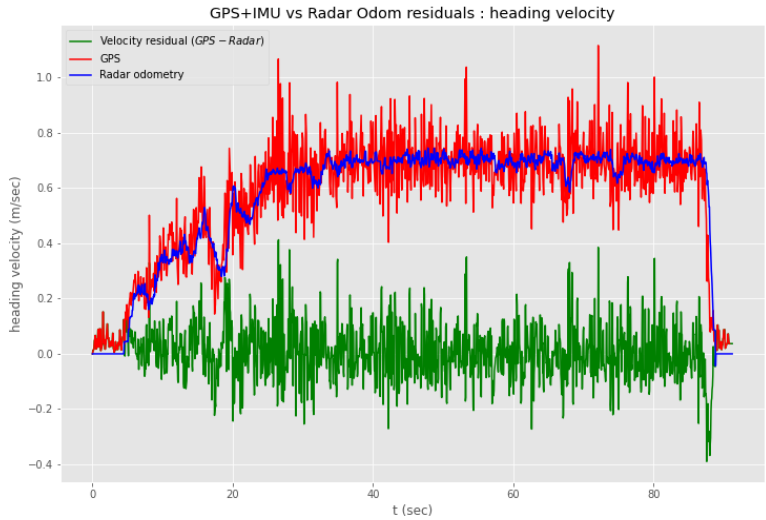


Figure 8.3: Radar odometry does a really good job at estimating heading velocity. However, when the vehicle hits the break (at the 85-second mark), some relatively large residuals can be observed.

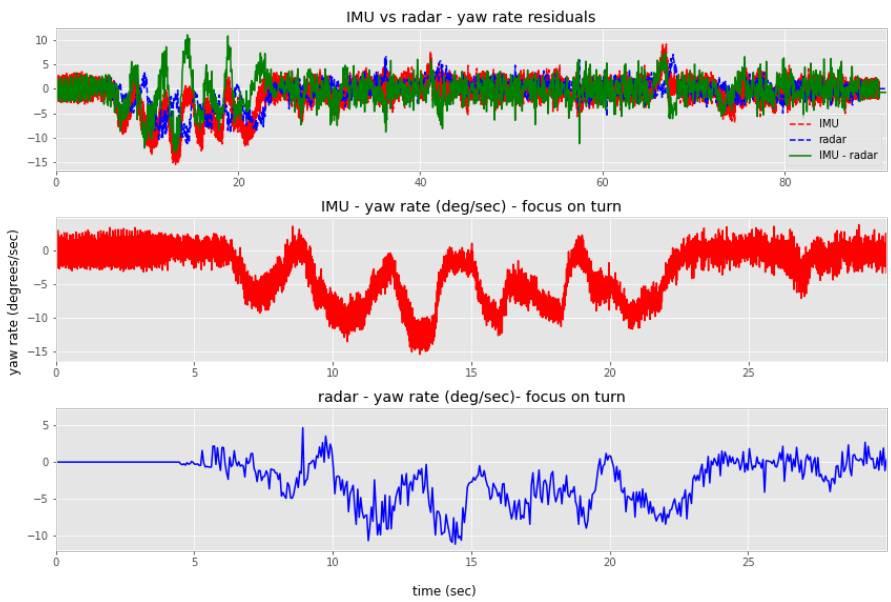


Figure 8.4: Radar odometry seemingly does a questionable job at estimating yaw rate. Especially when focusing on the residuals, one realizes that the results are not very accurate.

Figure 8.4 is not decisive enough when it comes to determining how reliable the yaw rate estimates are. This can be dealt with by integrating both the IMU and the radar odometry yaw rate estimates over time and comparing their yaw estimates. This can be seen in figure 8.5.

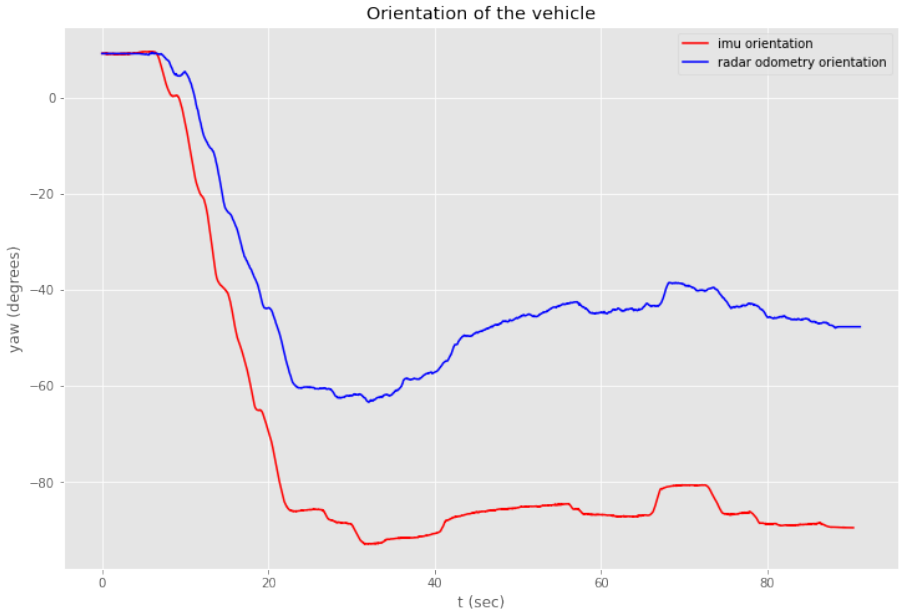


Figure 8.5: Radar odometry significantly underestimates the right turn that the vehicle takes (by roughly $20°$). This corresponds to an orientation root mean square error (RMSE) of 34.4637.

Figure 8.5 leads to the conclusion that the yaw rate estimates of the radar odometry algorithm cannot be trusted. Therefore, they will not be utilized for either the sensor fusion or the fault detection part of this thesis.

8.2 Sensor fusion

8.2.1 Simulated data

The validity of the implemented ESEKF algorithm was first assessed with simulated data. A 2d spiral trajectory was generated and the results of the ESEKF were observed (figure 8.6.

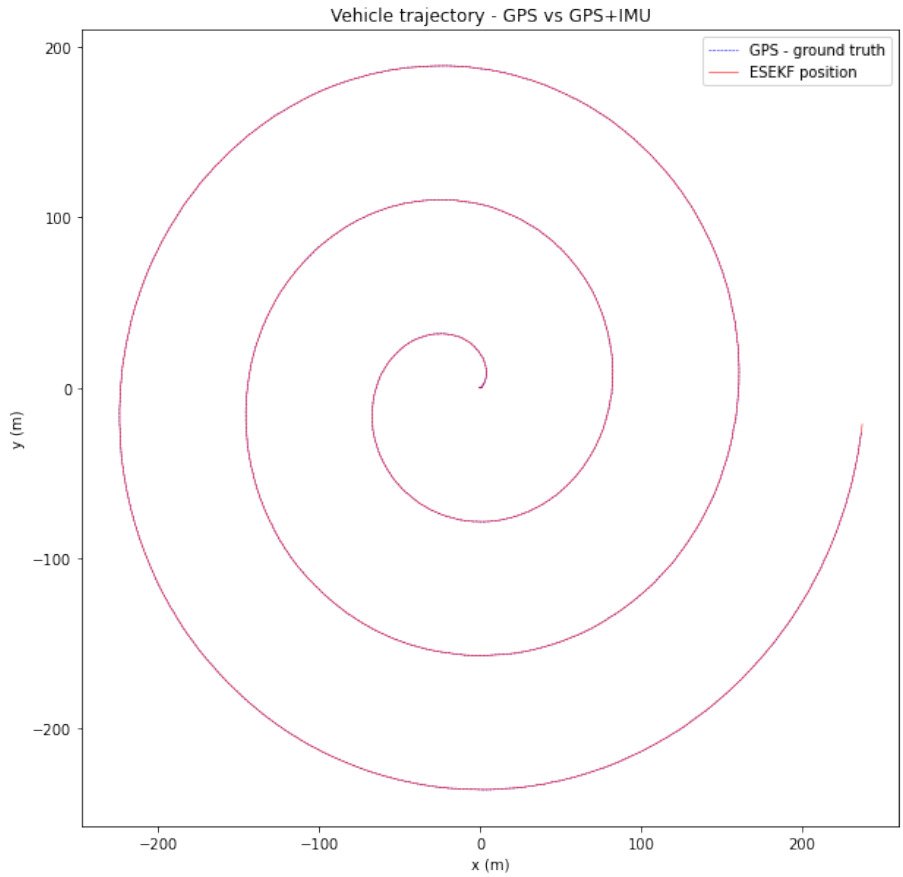


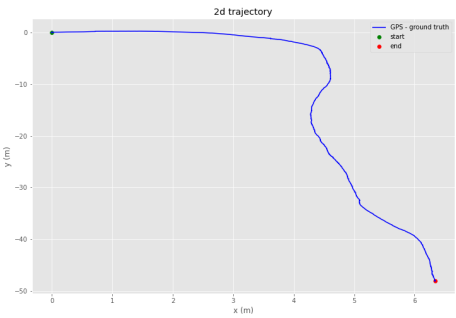
Figure 8.6: The ESEKF, as expected, perfectly follows the simulated trajectory.

8.2.2 Real data

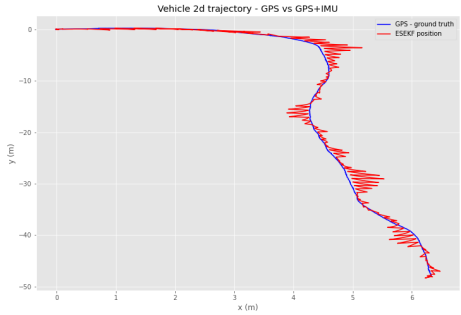
There are 3 possible sensor fusion combinations within the ESEKF that should be assessed:

1. Standard configuration
    - Prediction: IMU
    - Correction: GPS
2. Extended configuration
    - Prediction: IMU
    - Correction #1: GPS
    - Correction #2: Radar odometry
3. Dead reckoning
    - Prediction: IMU
    - Correction: Radar

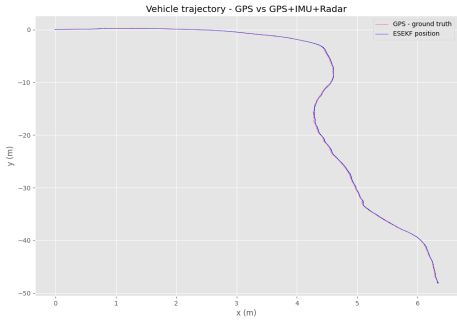
The results of the ESEKF for the above combinations can be seen in figure 8.7.



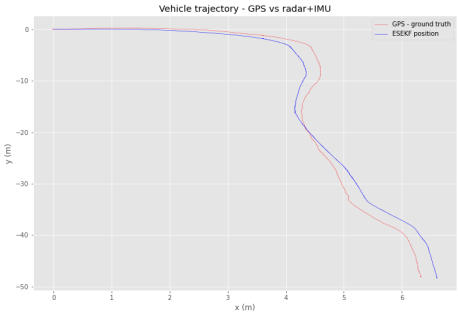
(a) The ground truth 2d trajectory, as measured by the GPS.



(b) IMU-based updates in the ESEKF enable navigation in-between GPS signals. However, because there is a large uncertainty in the linear acceleration measurements, the GPS corrections are essential.



(c) Radar-based corrections in the ESEKF mitigate the divergent from the ground truth position estimates, as the time period during which the ESEKF relies solely on the IMU is halved.

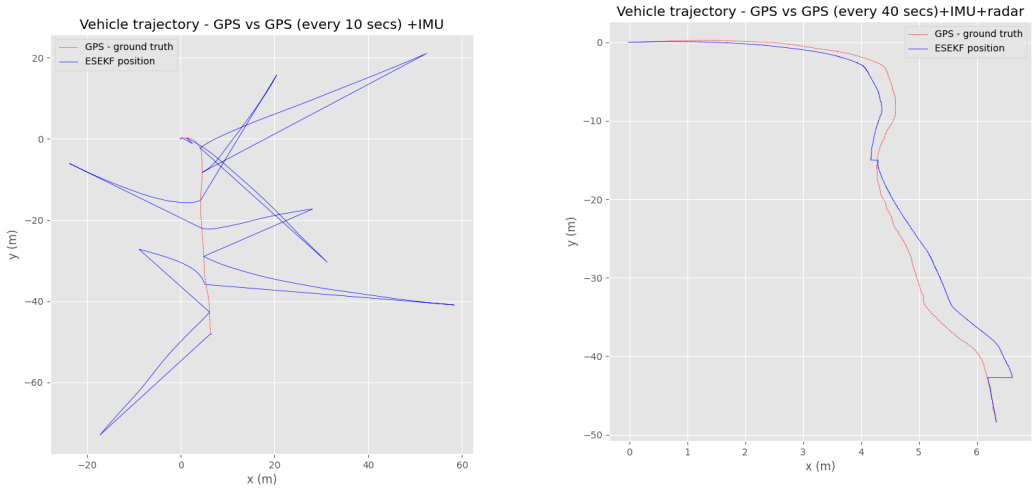


(d) Even when dead reckoning for the duration of the entire dataset (roughly 90 seconds), the vehicle manages to maintain a relatively accurate trajectory estimation with a translation RMSE of 0.1813.

Figure 8.7: The three different sensor fusion combinations trajectories, along with the ground truth trajectory.

8.2.3 GPS loss

It is essential, in order to evaluate radar odometry's contribution to the ESEKF, to investigate what happens when the GPS signal is not available. This is done by comparing the ESEKF 2d trajectory estimates when using the GPS and the IMU to what happens when the radar odometry estimates are also integrated. This is seen in figure 8.8.



(a) IMU linear acceleration measurements are unreliable, leading to very wrong trajectory estimations when the GPS signal gets lost for 10 seconds.

(b) Radar odometry heading velocity estimations coupled with the IMU angular velocity readings enable relatively good trajectory estimations, even when the GPS signal gets lost for 40 seconds.

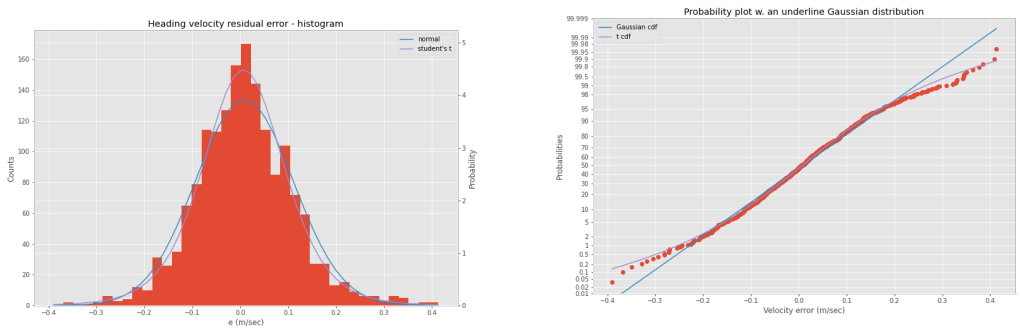
Figure 8.8: The effect that a loss in the GPS signal has on the ESEKF estimated 2d trajectory.

8.3 Fault detection

8.3.1 Offline residuals - statistical analysis

The underlying distribution p_0 of the GLRT, which corresponds to H_0, was found by fitting the offline heading velocity residuals $v_{res} = v_{gps} - v_{RO}$. Crucially, a distribution for which a closed form GLRT exists is needed. This is because the application that this thesis project is part of should run in real-time and using libraries like python's scipy for determining the MLE parameters adds a huge computational overhead. Specifically, when testing the Burr distribution with scipy's "fit" method, extracting the values of g within a single window of size 40 took 17 seconds. This is obviously too much, as consecutive radar samples arrive every 0.05 seconds.

In this context, the student's t distribution fitted the data adequately (fig. 8.9).



(a) Scipy's student's t distribution fits the heading velocity residuals quite well, having a steep "bell" shape around 0 and heavy tails.

(b) The underlying distribution of the probability plot is gaussian, which makes the gaussian PDF a straight line. Student's t seems to fit the data better in the upper than the lower tail.

Figure 8.9: Student's t distribution fits the data much better than the normal distribution.

8.3.2 GLRT assumptions

To test the first GLRT assumption from 5.3, one only needs to inspect figure 8.9, to realize that it does not hold entirely, as there are multiple samples that diverge from the student's t PDF.

The second assumption from 5.3 was tested with an auto-correlation plot (8.10) and found to be wrong as well.

This is not uncommon, as most real-life application do not fully comply with the two theoretical assumptions of the GLRT.

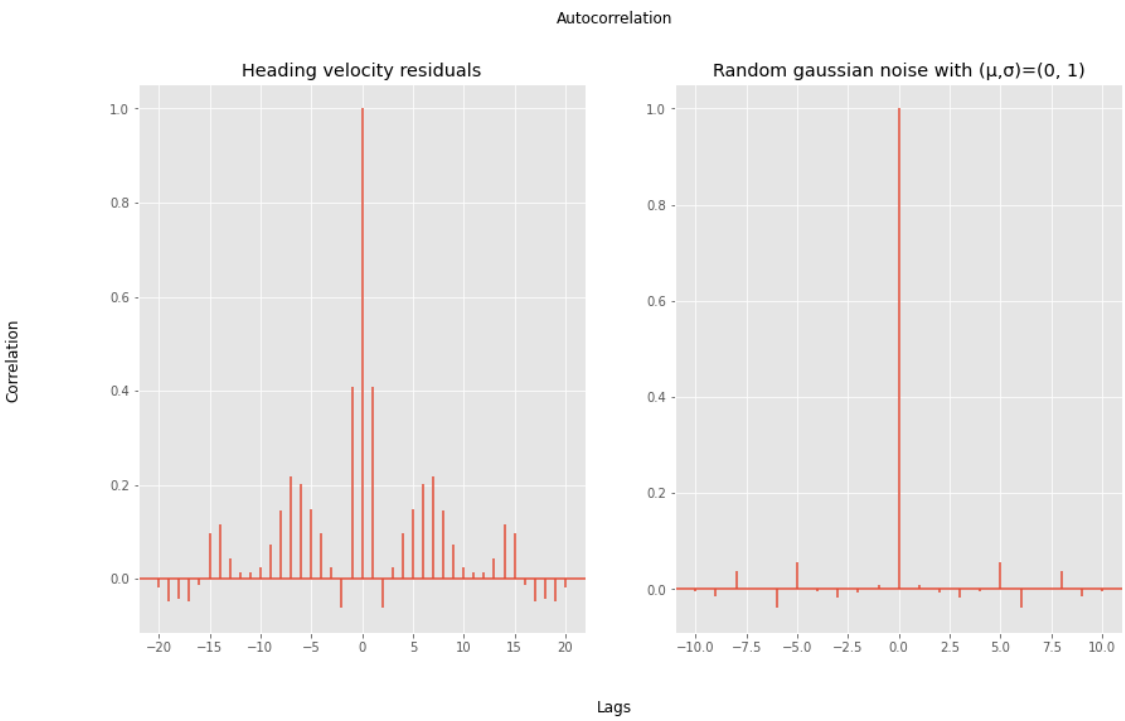


Figure 8.10: There is a very large correlation between the drawn residual and the ones within one lag. Even further away, auto-correlation is significant.

To test the third assumption from 5.3, three different GPS faults were introduced to a window of the velocity residuals from the offline statistical analysis. The size of that window was 40 and the faults were similar to the ones in section 5.4. Consecutively, the mean, standard deviation and statistical parameters of the student's t distribution that best fits the new sequence of residuals were compared to the ones form the original sequence.

It can be seen in figure 8.11 that not only the mean, but all the inspected parameter values of the sequence change after the fault has been introduced. This means that the third GLRT assumption doesn't fully hold as well. That said, this third assumption is also quite common and it is the same assumption that was made by [Willersrud et al. 2015].

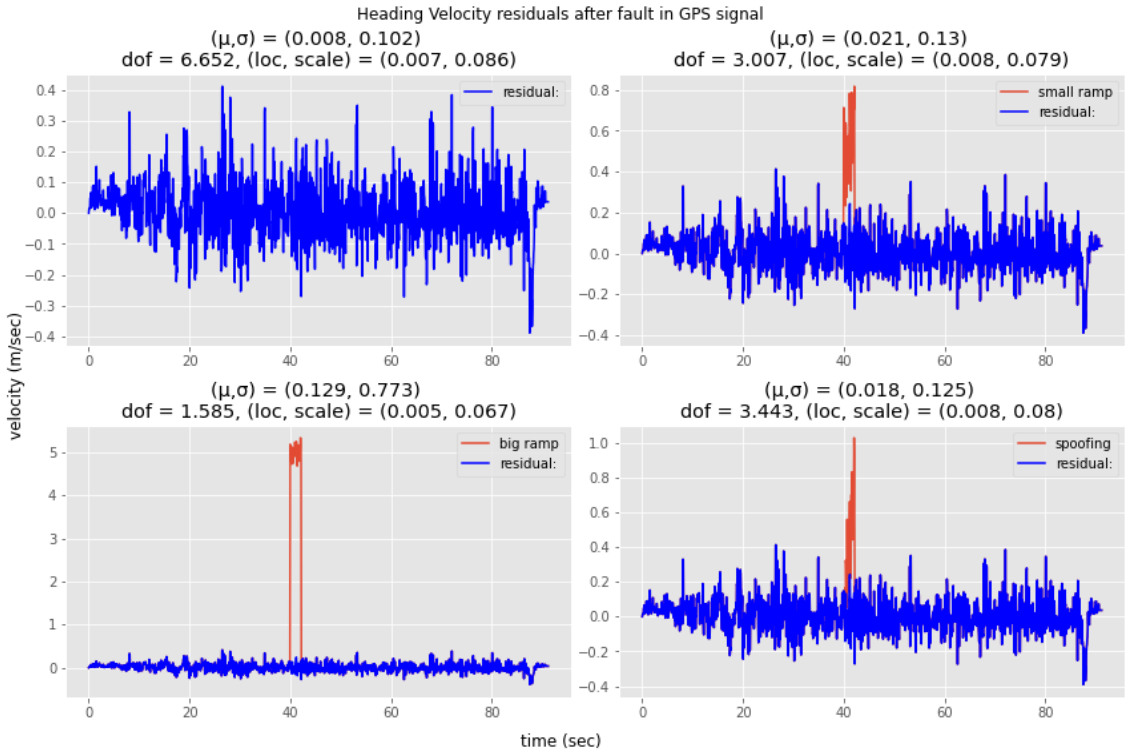


Figure 8.11: The introduced errors to the residuals sequence clearly alter not only the mean of p_0, but also the standard deviation σ_0.

Regarding figure 8.11, several notes need to be made:

- The mean parameter changes notably under all three inspected faults.
- The (loc, scale) parameters only slightly change compared to the rest.
- The standard deviation value only changes significantly in the case of a ”big ramp” error. This is good, because ”big ramp”-type errors cal also be detected with alternative, simpler methods. For example, one could simply monitor the GPS velocity and acceleration estimates and flag the ones that seem unnatural (larger than the maximum allowed values for the vehicle within the environment that it operates).
- The DOF parameter, unfortunately, varies significantly. This is an issue, as the implemented GLRT assumes that it will remain stationary.

The fact that the above assumptions don’t hold entirely does not mean that the GLRT can not still be used. However, it does mean that the detector will be suboptimal and the false alarm rate higher than expected.

8.3.3 Window size determination

The window size w must be determined taking into account the window size properties from 5.5 and the specific requirements of the application this thesis project is aimed for. This subsection will clarify why this thesis project selected a window of size 40. The clarification will be based on the aforementioned properties.

Property #1: Smaller window sizes increase the probability of falsely detected faults.
The window size should be as large as possible, whilst ensuring that the probability of false alarm remains as defined in 8.3.4.

Property #2: Larger window sizes increase the ”detection time”.
Given that the top speed of the vehicle is 2 m/sec and the fact that the margin of positioning

error is 2 m (both given by the collaborating company), a fault should not be detected later than 1 second after it occurs.

Property #3: Larger window sizes add a significant computational overload, which could render the execution of the algorithm in real-time unfeasible.
Given that radar scans arrive every 0.05 seconds and the fact that other tasks might need to be executed by the vehicle processor within this time period, a window size that results in estimation of $g(k)$ values in less than 0.05 seconds needs to be selected. However, testing is performed in python, whereas the ROS2 node that will be executed in real-time by the robot will be in C++. Since C++ is much faster than python, the 0.05 limit could be approached relatively safely. In all regards, window sizes that lead to test estimations of $g(k)$ in over 0.05 seconds are ruled out.

Property #4: Larger window sizes increase the ”recovery time”.
This means that selecting a very large window size leads to the vehicle detecting a fault in the monitored sensor long after the fault has stopped existing. On the one hand this is not extremely important because false positive fault detections are not as dangerous as false negative ones within autonomous agricultural applications. On the other hand, if the vehicle relies on an auxiliary positioning system, this will be bound to accumulating drift. This, in turn, means that the longer the vehicle operates without the ”detected as faulty” GPS, the larger the position uncertainty becomes, which is unsafe. Ergo, too large window sizes should be avoided.

For the purposes of this project, the window sizes of 20, 40 and 100 were tested. These correspond to windows with time periods 1, 2 and 5 seconds respectively. The results in terms of the GLRT test statistic $g(k)$ can be seen in figures 8.12, 8.13 and 8.14.

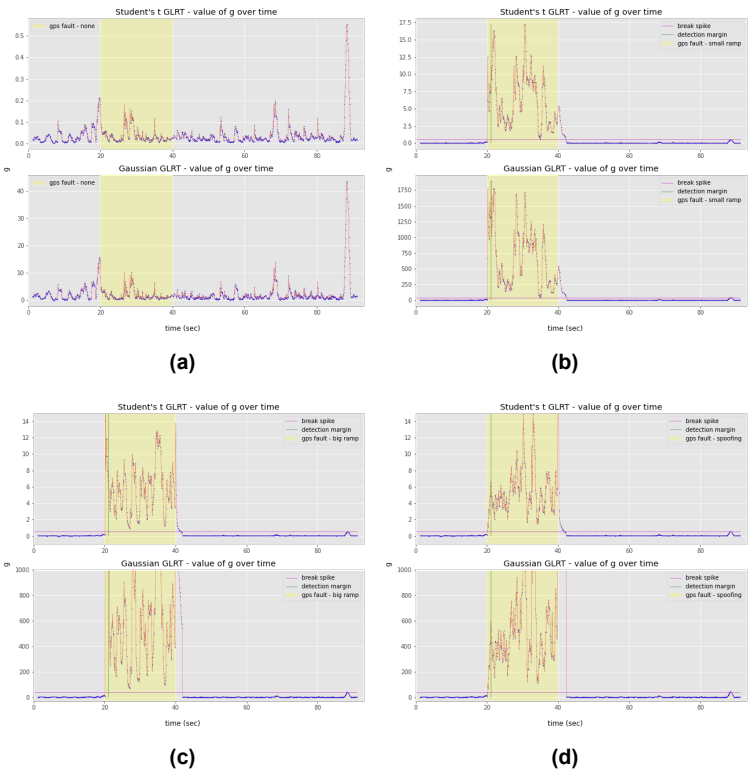


Figure 8.12: The returned GLRT test statistic $g(k)$ values with window of size 20.

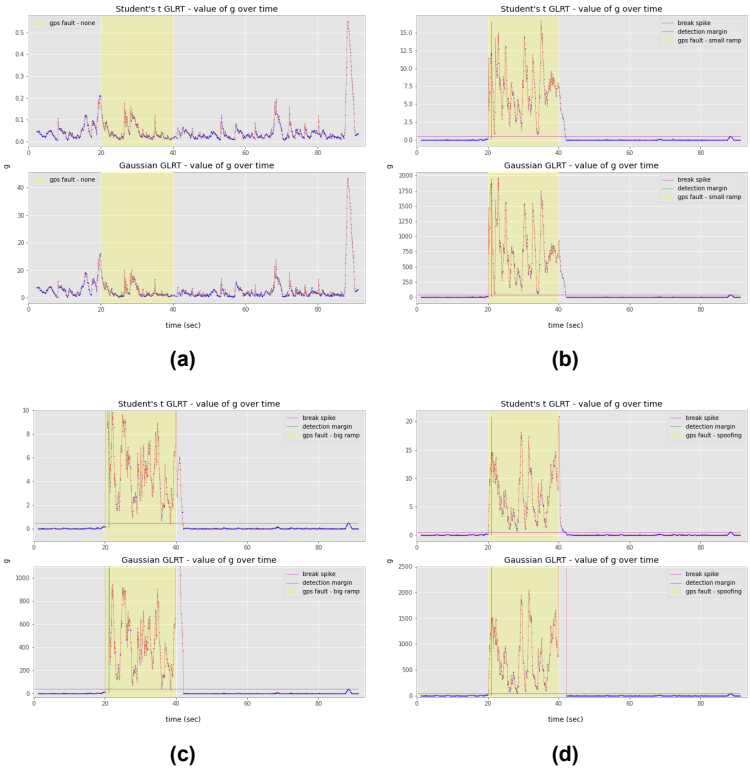


Figure 8.13: The returned GLRT test statistic $g(k)$ values with window of size 40.

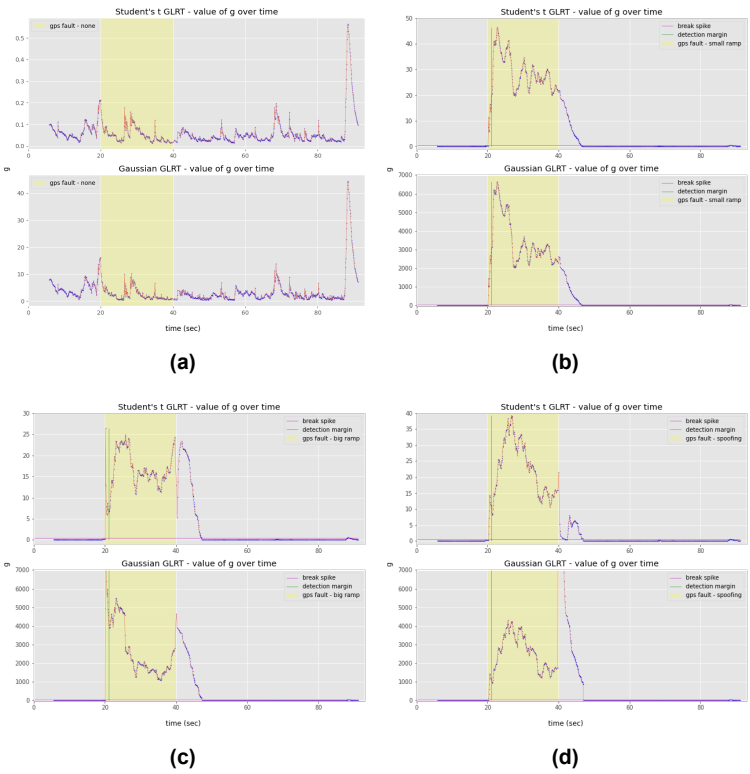


Figure 8.14: The returned GLRT test statistic $g(k)$ values with window of size 100.

Having experimented with the three different window sizes and after observing the figures above, the following remarks were made:

1. $W = 20$ is too small, as it led to very low g values during the GPS fault. These could be mistaken with values of g when there is no fault in the GPS sensor. This is especially visible at the 90-second mark, when the vehicle hits the break, which leads to temporarily higher g values. This is likely to lead in a lot of false alarms and therefore, $w = 20$ is excluded. This ties directly to property #1.
2. In all three cases, the fault seems to get detected after much less than 1 second, so property #2 is not an issue.
3. When it comes to property #3, $w = 100$ led to approximately 0.05 seconds of estimation time for each $g(k)$. This is the upper limit. Therefore, higher values of w should not be considered.
4. Regarding property #4, after the GPS fault had disappeared w=100 lead to almost twice as much passed time compared to w=40 before the $g(k)$ values returned to normal values for all three investigated faults (~ 6 seconds, compared to ~ 3 seconds). Because of this, $w = 100$ is excluded.

The above remarks led to picking $w = 40$ as the window size. Moreover, the following observations seem particularly interesting and worth noting:

- The gaussian and t distribution GLRTs seem to result in similar patterns for the timeseries of the generated values of $g(k)$. The major difference between the two seems to be the scale of the values. Ergo, there is no visible advantage in keeping both of them for the next step of fault detection. Because the t distribution fitted the velocity residuals much better, this is the one that will be kept.
- The $g(k)$ values are wildly and unpredictably fluctuating. This is also true for the time period during right after the GPS fault has disappeared (figure 8.14d). To combat this, a second (falling) threshold will be added, which would be smaller and replace the first (rising) threshold after a fault has already been detected.

8.3.4 Threshold(s) determination

Rising threshold

The first step in threshold determination, as mentioned in 5.2, is determining the desired probability of false alarm (PFA). This is done by deciding how often it is acceptable to raise a false alarm. That, after communication with the collaborating company, was set to be 30 days. This corresponds to a PFA of 0.00000001929.

The next step is to analyze the outputs $g(k)$ of the GLRT and the distribution that best fits them. This, for $w = 40$, can be seen in figure 8.15.

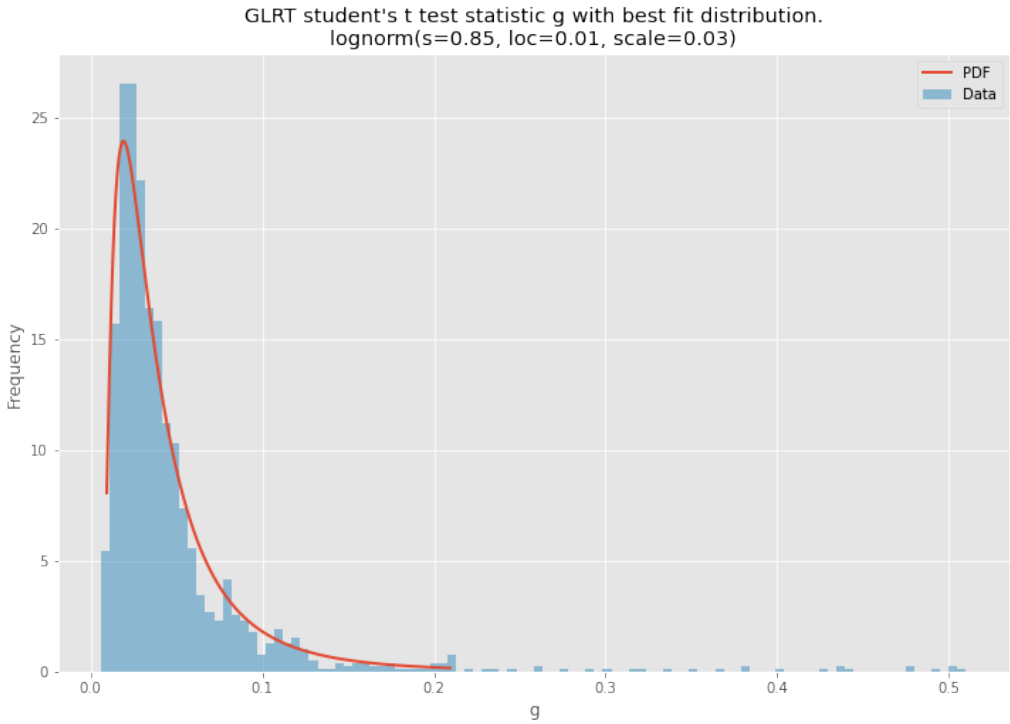


Figure 8.15: Scipy's lognorm PDF best fits the student's t GLRT g values for $w = 40$.

Threshold determination through the CDF plot can be seen in figure 8.16.

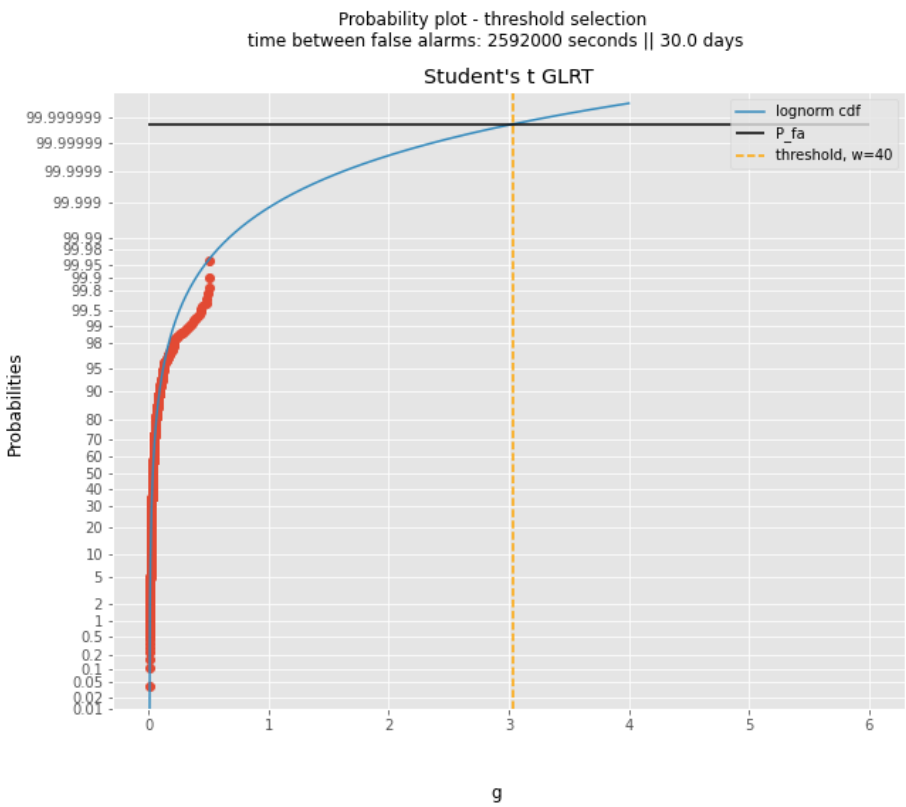


Figure 8.16: The fault detection thresholds emerge from the theoretical CDF of g and depend on the distribution choice, the determined PFA, and the window size w.

From figure 8.16 it also becomes visible how larger window sizes lead to higher thresholds, which leads to lower chances of detecting a false alarm. Finally, it becomes apparent that the gaussian and student's t GLRT operate with completely different scales of the GLRT test statistic g.

The rising threshold value for the student's t GLRT was found to be 3.037.

Falling threshold
The falling threshold is arbitrarily set to a fraction of the found rising threshold. The lower the falling threshold, the less likely it is that g fluctuations will lead to false positive and false negative alarms. After experimentation, it was found that g values always return to values less than $\frac{1}{10}th$ of the rising threshold, while fluctuations never lead to so small values. In this sense, the falling threshold was set to 0.3037.

8.3.5 Trajectories under fault

The determined parameters of the GLRT can be seen in table 8.1.

	GLRT
Distribution	Student's t
Test statistic $g(k)$ distribution	Lognorm
Probability of false alarm	0.00000001929
Window size	40
Rising threshold	3.037
Falling threshold	0.3037

Table 8.1: The parameters of the designed GLRT.

Having determined the parameters of the GLRT, the next step is to perform fault detection on the training dataset under the simulated GPS errors and evaluate the resulted trajectories. This can be seen in figure 8.17.

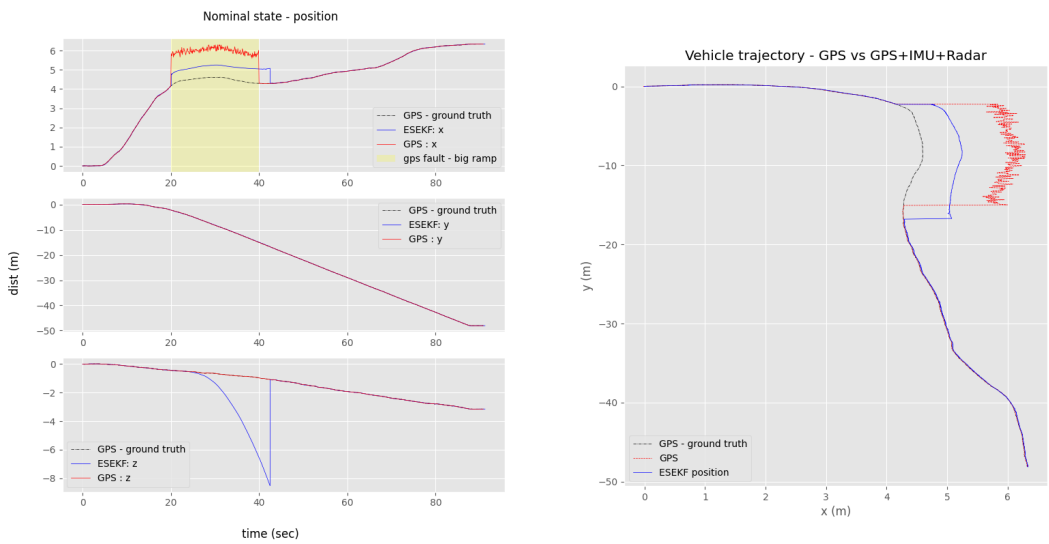


Figure 8.17: Position and trajectory of the vehicle under a "big ramp" error and fault detection with a student's t GLRT.

Figure 8.17 indicates a serious problem with fault detection as it is. This originates in the fact that it takes a couple of GPS signal returns before the test statistic rises above the

rising threshold. If this is not dealt with, the initial position of the vehicle when the auxiliary system takes over will be faulty, which will make the entire predicted trajectory during the fault wrong as well.This can be dealt with in two ways:

1. Replace the last GPS reading with the GPS reading x samples ago. This will be referred to as ”rollback”, from now on.
2. Remove the last few GPS readings and start the auxiliary localization from the GPS position x samples ago. This will be referred to as ”advanced rollback”, from now on.

”Rollback” is much easier to implement, codewise, but ”advanced rollback” is a more complete solution that utilizes all the available sensor information. That said, because the spike above the rising threshold in the test statistic during the training dataset happens after only one faulty GPS return, ”rollback” works indistinguishable well. This is because, given that GPS signals arrive at a frequency of 1 return per second, even at the top speed of 2 m/sec, the vehicle will only ”forget” 20 cm of travelled distance. Therefore, this thesis implemented the simpler ”rollback” technique. The results of fault detection with ”rollback” can be seen in figures 8.18, 8.19 and 8.20.

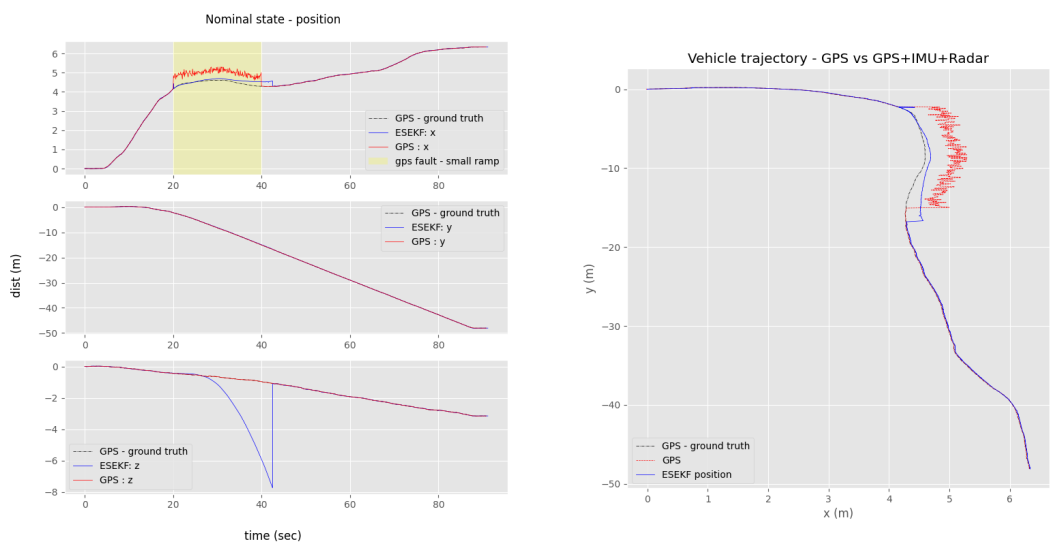


Figure 8.18: Position and trajectory of the vehicle under a ”small ramp” error and fault detection with a student’s t GLRT.

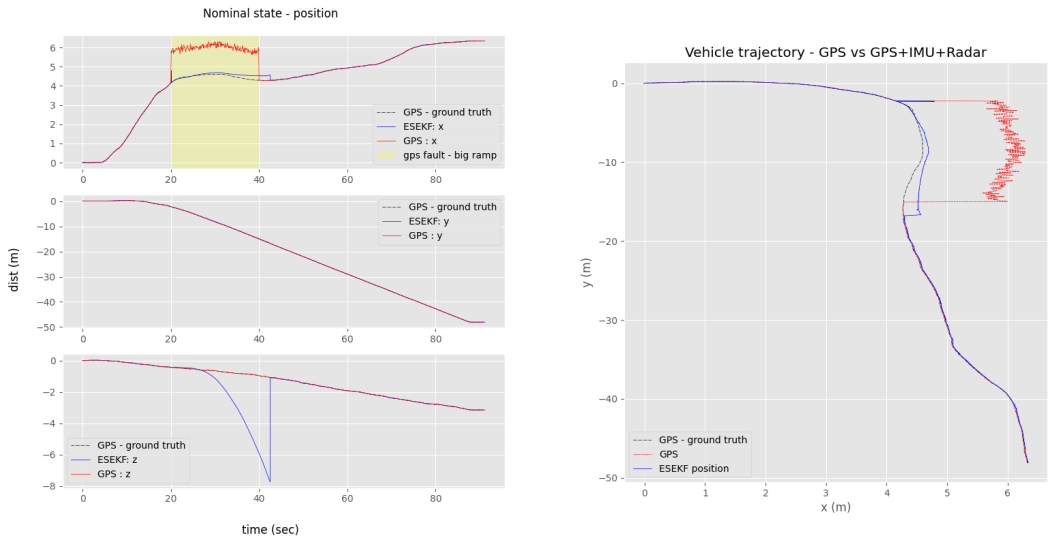


Figure 8.19: Position and trajectory of the vehicle under a ”big ramp” error and fault detection with a student’s t GLRT.

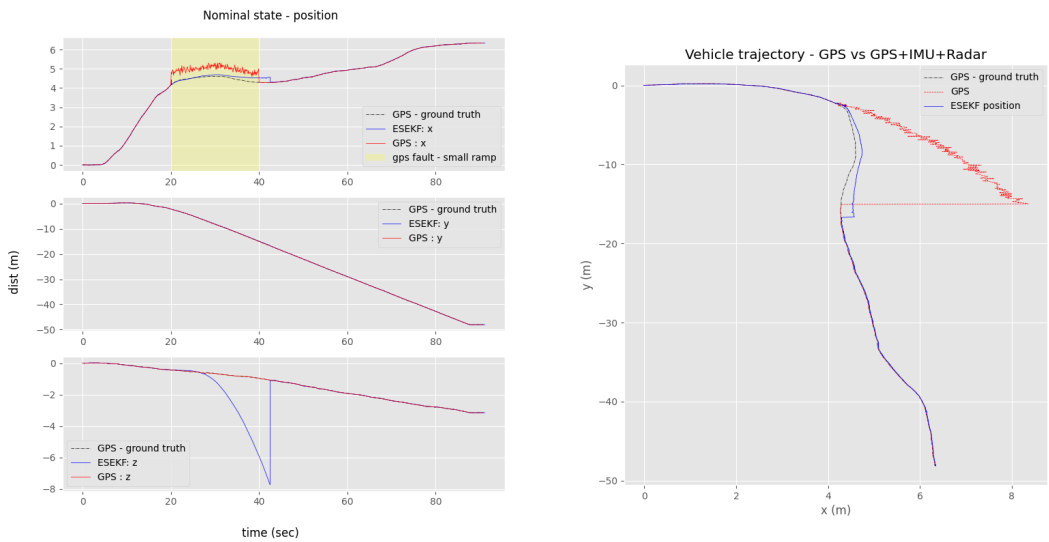


Figure 8.20: Position and trajectory of the vehicle under a ”spoofing” error and fault detection with a student’s t GLRT.

The above figures (8.18, 8.19, 8.20) show that the GPS fault was detected in all three cases and during the time period of the fault the vehicle was able to localize itself with the auxiliary system (RO + IMU). The small visible position spikes right after the fault arises indicate the time instance when the ”rollback” technique was applied.

9 Validation

This section contains the results of the validation process. This was a two-fold procedure. The first step was to validate that the transferred ROS2/C++ code worked the same way as the initial python code. The second step was to test the performance of the implemented algorithms in the validation dataset in ROS2/C++.

9.1 ROS2/C++

9.1.1 Radar ego-motion

The radar ego-motion performance in terms of the heading velocity estimates is extremely similar to the one coming from the training dataset in python. This can be seen in figure 9.1, when compared to figures 8.2b and 8.3. The radar odometry yaw rate estimates were ignored, as it was proved during the training dataset that they are inaccurate.

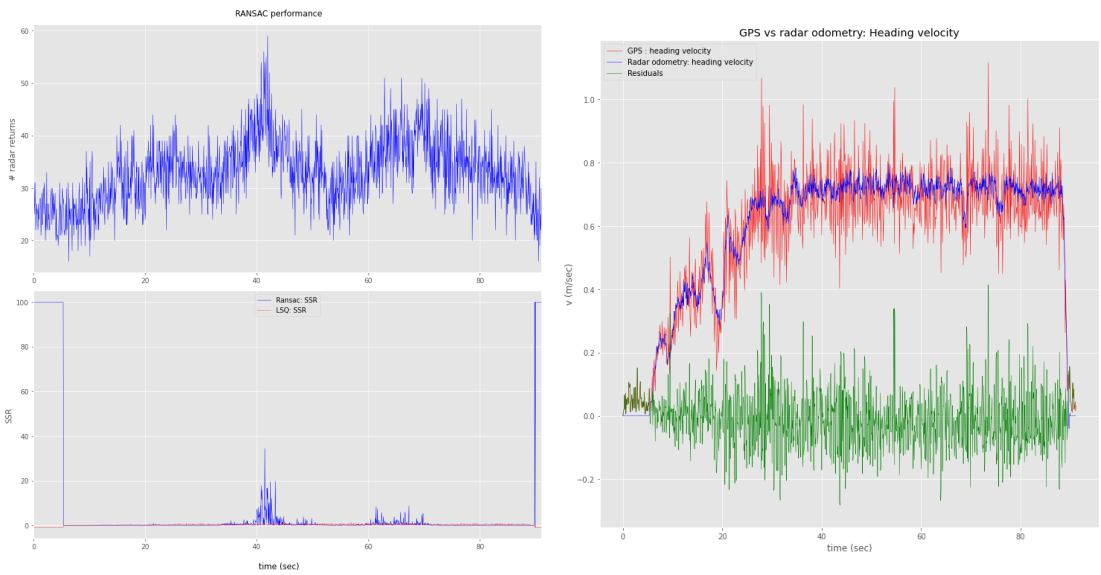
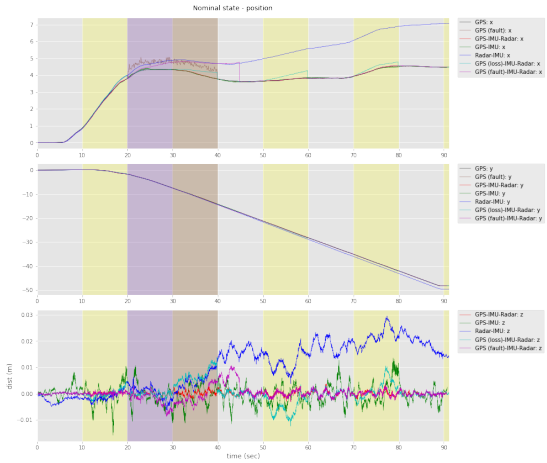


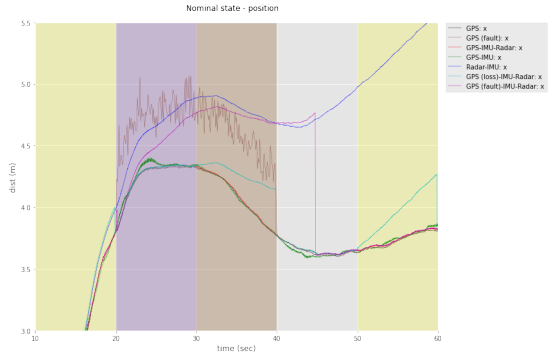
Figure 9.1: Heading velocity estimation performance of the radar ego-motion algorithm on the training dataset in ROS2/C++.

9.1.2 Sensor fusion and fault detection

The position estimates of the ESEKF in ROS 2/C++ can be seen in figures 9.2 and 9.3. The injected GPS fault was a ”small ramp”.



(a) Position estimations by the ESEKF with different sensor fusion combinations.



(b) Figure 9.2a zoomed in the area of interest.

Figure 9.2: The position estimates of all the different ESEKF sensor fusion combinations.

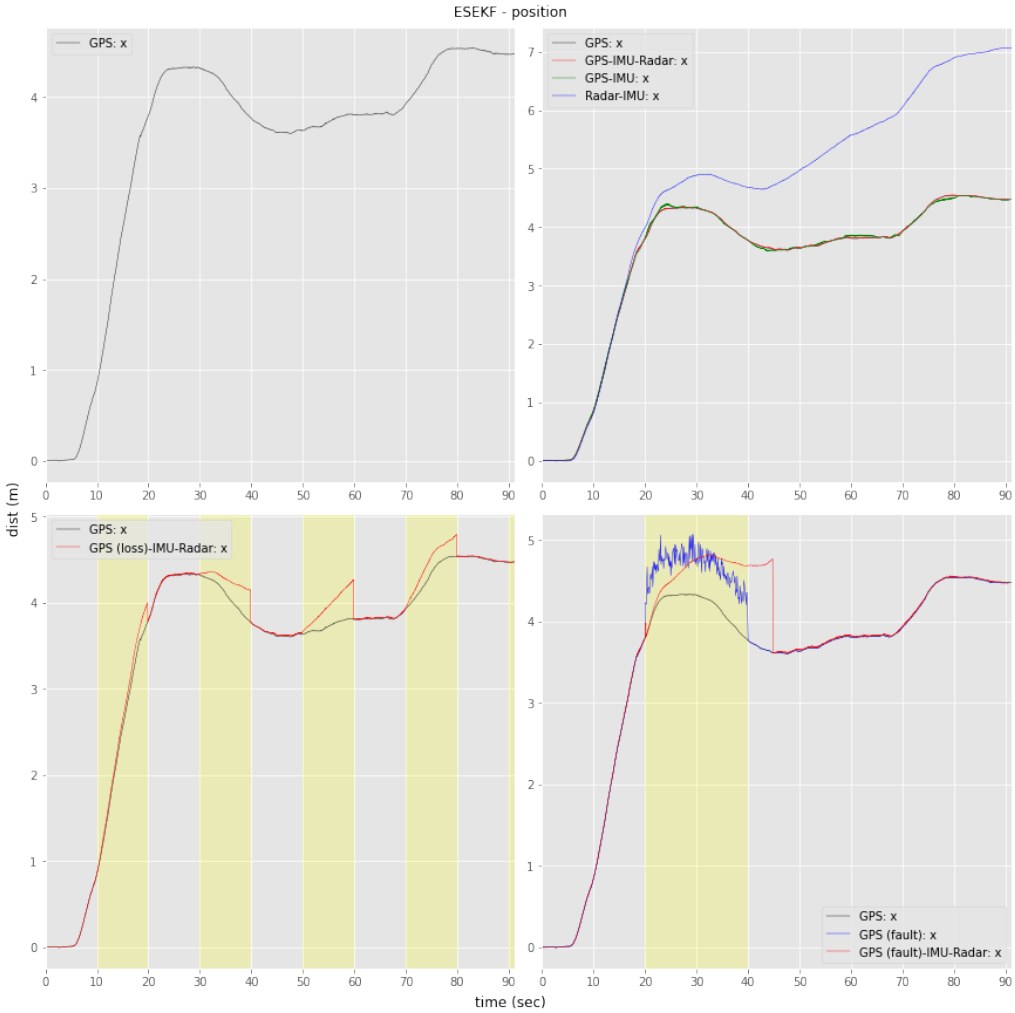


Figure 9.3: The x position estimates of the ESEKF. The effect that possible GPS losses or GPS faults have on the 2d trajectory estimation is made especially visible in the bottom left and right figures.

The 2d trajectory estimates of the ESEKF in ROS 2/C++ can be seen in figure 9.4.

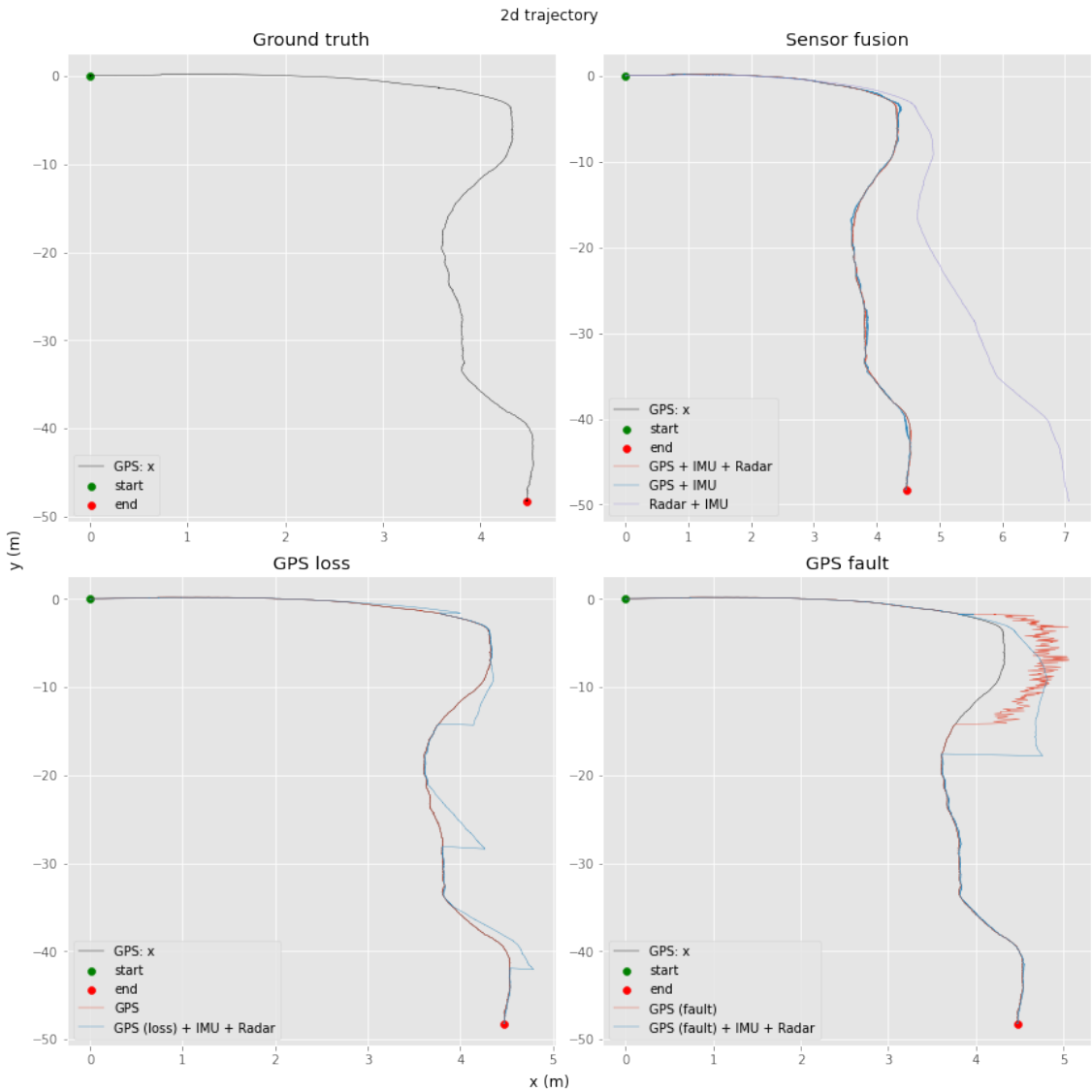


Figure 9.4: The 2d trajectory estimates of the ESEKF. The dead reckoning case (top right) results in a translation RMSE of 1.1460.

It is apparent that the estimated position and therefore, the trajectory of the vehicle while dead reckoning for the entire training dataset is different than the one that resulted from the python tests (figure 8.7d). This is likely caused by the ”orientation issue” mentioned in section 6.2 combined with the fact that the python training dataset and the ROS 2 bag have slightly different starting timestamps, which affects the estimates as well. That said, it is still reasonable to conclude that the ROS2 nodes work well, as the results are not that far off and the ROS 2 estimated trajectories show similar patterns. For example, the GPS+IMU fusion still follows the GPS ground truth perfectly and the radar+IMU fusion results in a trajectory with a similar shape and some gradually accumulating drift.

9.2 Validation dataset

9.2.1 Radar ego-motion

The radar ego-motion heading velocity estimates on the validation dataset in ROS 2/C++ are bad.

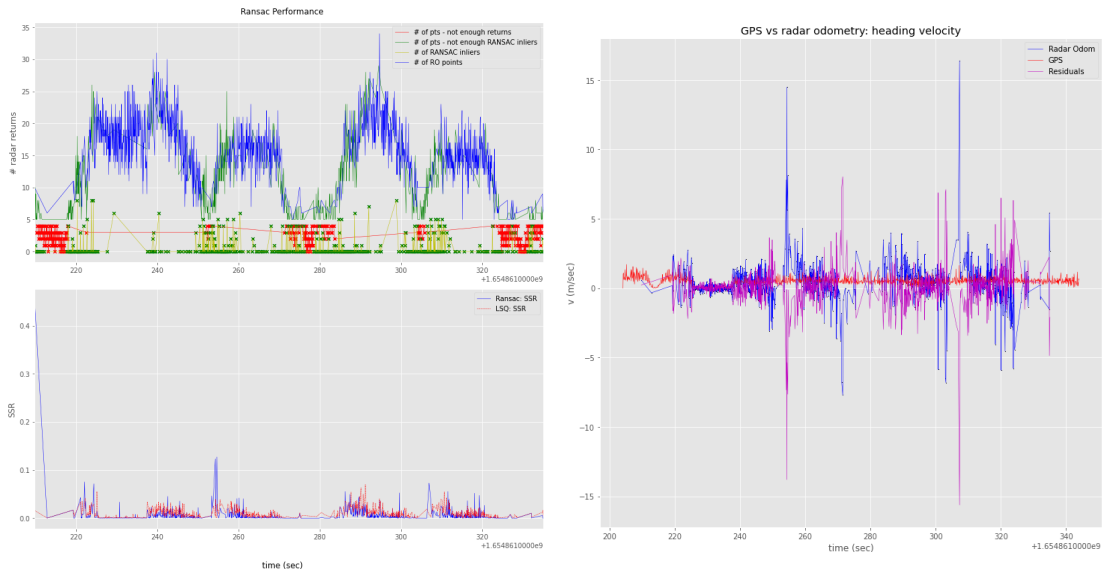


Figure 9.5: Heading velocity estimation performance of the radar ego-motion algorithm on the validation dataset in ROS2/C++.

Taking a closer look at the left subfigure of figure 9.5, two important notes can be made:

- The validation dataset includes too small radar pointclouds.
- Even when larger pointclouds are returned by the radar it is often the case that RANSAC cannot find a velocity profile that matches them to remove outliers.

Apart from these, the importance of the tuneable parameters described in section 3.5 becomes visible. For example, a smaller number of fewest acceptable points/inliers would have led to more inliers. At the same time, it is very possible that a different velocity error tolerance would have significantly decreased the worrisome number of times RANSAC was not able to find a single inlier.

These poor RO velocity estimates are a major problem, as the sensor fusion and fault detection parts of this thesis depend on them. This is to say that the results from the following subsection are expected to behave poorly.

9.2.2 Sensor fusion and fault detection

Since the validation dataset, as mentioned in 6.2, includes orientation measurements, an orientation correction update was also added to the ESEKF of the validation set.

The orientation, position and trajectory estimates from the validation set in ROS 2/C++ can be seen in figures 9.7 and 9.8.

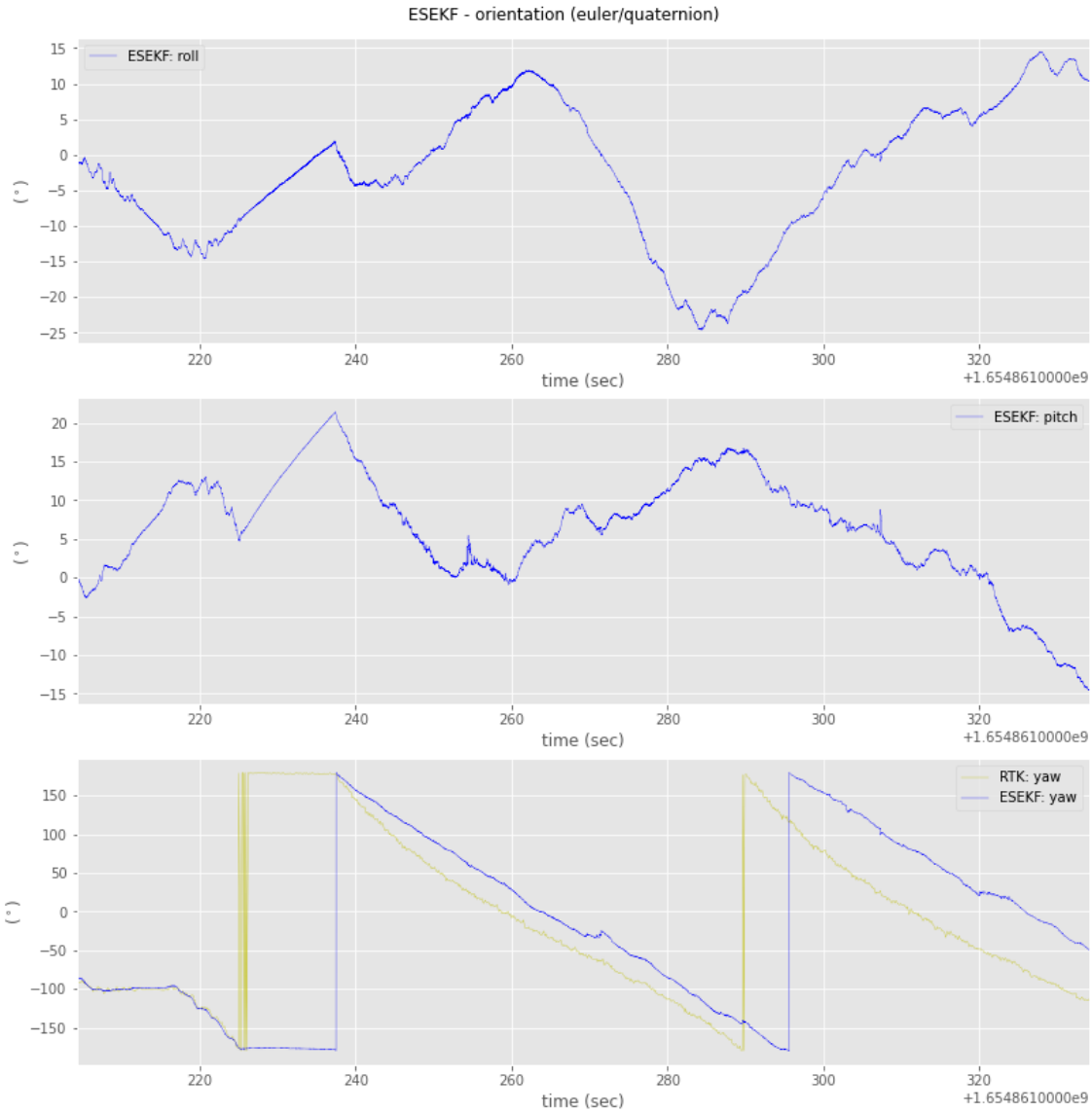


Figure 9.6: Orientation estimates of the ESEKF. There is an unexplained lag between the IMU estimate and the RTK antenna estimate.

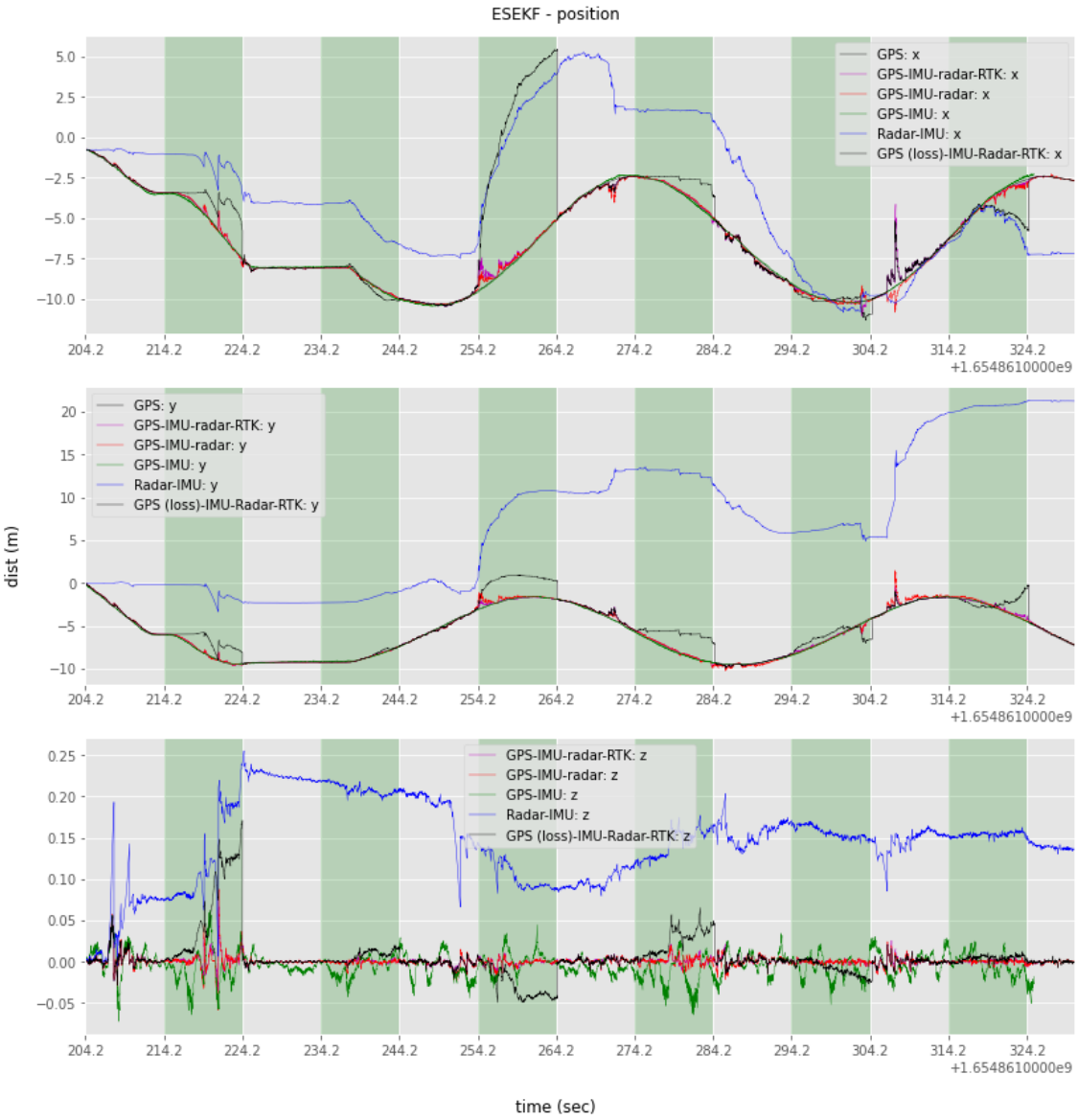


Figure 9.7: Position estimates of the ESEKF. Dead reckoning with radar and IMU is unreliable.

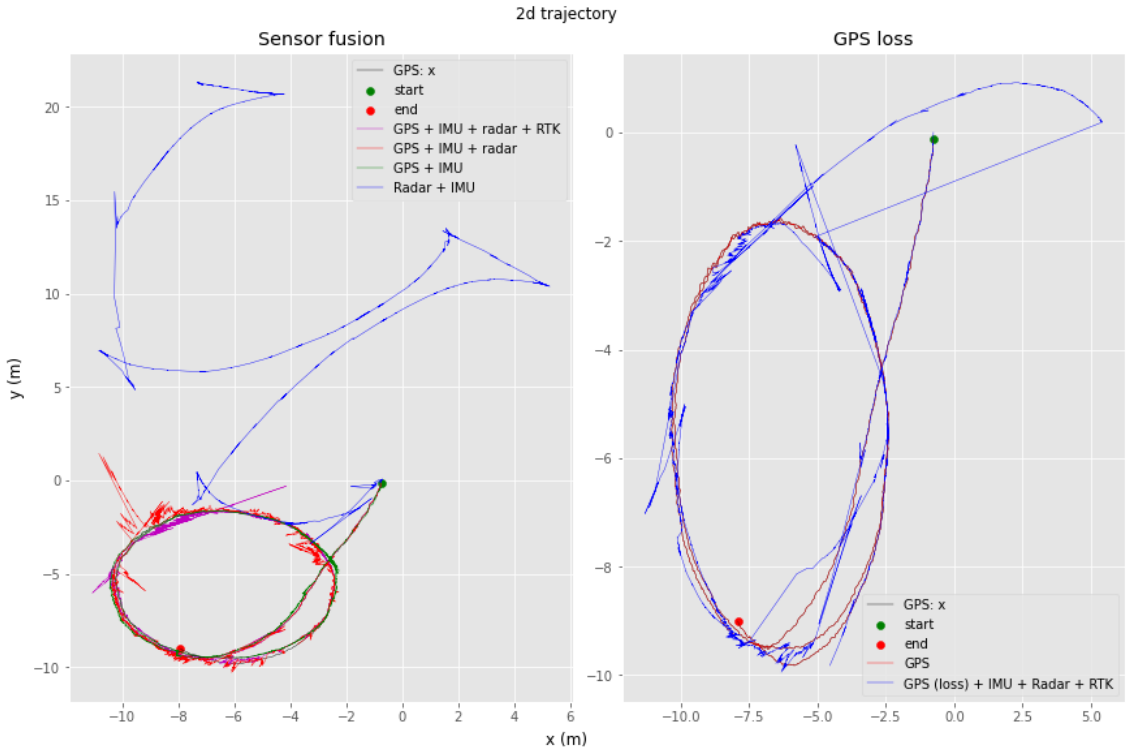


Figure 9.8: The 2d trajectory estimates of the ESEKF. Dead reckoning is unreliable, while adding radar odometry to the standard GPS+IMU fusion only makes the trajectory estimation worse. The RTK orientation measurement significantly changes the trajectory estimation.

There is no need to test the GLRT in the validation set. This is because the radar odometry heading velocity estimates are so poor (figure 9.1) that the GLRT test statistic values would constantly be both wildly fluctuating and much higher than the chosen rising threshold. Therefore constant false alarms would be raised. During the periods of these alarms the estimated vehicle trajectory would be completely off.

10 Conclusions

This thesis attempted to perform fault-tolerant localization in an agricultural environment using GNSS, IMU and radar systems.

The algorithm proposed by [Kellner et al. 2013] is found to have excellent results on the training dataset in terms of heading velocity estimation. On the other hand, the yaw rate estimation is sub-par. Large radar pointcloud returns and a stationary environment affect the algorithm's performance significantly.

The ESEKF as described by [Sola 2017] shows great results in fusing the signals from the used sensors. Low-quality correction measurements can notably worsen the state estimates.

Given a high quality heading velocity estimation by the radar odometry algorithm, the student's t GLRT as proposed by [Willersrud et al. 2015] is able to detect three different kinds of faults and signal an alarm for the auxiliary positioning system to take over the localization task. The auxiliary system is able to provide dependable localization for small periods of time under one condition. This condition is that the orientation of the state vector is very accurate.

Generalizing the aforementioned results on different datasets and vehicle setups is tricky. The validation set that was used by this thesis showed terrible results in estimating the RO heading velocity and therefore, the auxiliary 2d trajectory. Several educated assumptions can be done as to why this happened, with the first one out of the following being by far the most important:

- The returned radar pointclouds are much smaller (figure 9.5).
- The radar placement on the validation vehicle is different (front-right looking diagonally compared to front-center looking straight ahead).
- There are many more occasions where moving objects block the radar's field of view (figures 6.6c and 6.6d).
- Possible timestamp errors in the ROS 2 bags. This suspicion is raised mainly because of figure 9.6.

In summary, completely autonomous localization using a non-GPS approach was not possible, whereas partial autonomous localization was. Providing fault-tolerance to the existing GPS-based localization method was possible. Validation was unsuccessful.

As future work, this thesis proposes the following:

- Multiple radars could be combined in the RO phase. This will lead to larger pointcloud returns and, if chosen, to a 3DOF velocity estimation (Kellner et al. 2014).
- A more sophisticated approach for performing orientation corrections could be applied in the ESEKF, as the current one naively replaces the state orientation with the measured one.
- A third means of extracting heading velocity estimations is added to the fault detection part, which will enable a three-way voting system for determining which sensor is faulty. This could be a VO system or an additional RO system.

Bibliography

Bhatti, Umar Iqbal and Washington Yotto Ochieng (2007). “Failure modes and models for integrated GPS/INS systems”. In: *The Journal of Navigation* 60.2, pp. 327–348.

Du, Yujun et al. (2021). “Vulnerabilities and integrity of precise point positioning for intelligent transport systems: overview and analysis”. In: *Satellite Navigation* 2.1, pp. 1–22.

Barnes, Dan and Ingmar Posner (2020). “Under the radar: Learning to predict robust keypoints for odometry estimation and metric localisation in radar”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 9484–9490.

Kellner, Dominik et al. (2013). “Instantaneous ego-motion estimation using doppler radar”. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, pp. 869–874.

Yokoo, Kaoru, Stephane Beauregard, and Martin Schneider (2009). “Indoor relative localization with mobile short-range radar”. In: *VTC Spring 2009-IEEE 69th Vehicular Technology Conference*. IEEE, pp. 1–5.

Fischler, Martin A and Robert C Bolles (1981). “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6, pp. 381–395.

Norris, William (1906). *Modern steam road wagons*. Longmans, Green, and Company.

Kellner, Dominik et al. (2014). “Instantaneous ego-motion estimation using multiple Doppler radars”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1592–1597.

Barjenbruch, Michael et al. (2015). “Joint spatial-and doppler-based ego-motion estimation for automotive radars”. In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, pp. 839–844.

Rapp, Matthias et al. (2015). “A fast probabilistic ego-motion estimation framework for radar”. In: *2015 European Conference on Mobile Robots (ECMR)*. IEEE, pp. 1–6.

Ghabcheloo, Reza and Shadman Siddiqui (2018). “Complete odometry estimation of a vehicle using single automotive radar and a gyroscope”. In: *2018 26th Mediterranean Conference on Control and Automation (MED)*. IEEE, pp. 855–860.

Doer, Christopher and Gert F Trommer (2020). “Radar inertial odometry with online calibration”. In: *2020 European Navigation Conference (ENC)*. IEEE, pp. 1–10.

Kramer, Andrew et al. (2020). “Radar-inertial ego-velocity estimation for visually degraded environments”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 5739–5746.

Narula, Lakshay, Peter A Iannucci, and Todd E Humphreys (2020). “Automotive-radar-based 50-cm urban positioning”. In: *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE, pp. 856–867.

Monaco, Chris D and Sean N Brennan (2020). “Radarodo: Ego-motion estimation from doppler and spatial data in radar images”. In: *IEEE Transactions on Intelligent Vehicles* 5.3, pp. 475–484.

Gao, Pengen et al. (2021). “Accurate Automotive Radar Based Metric Localization with Explicit Doppler Compensation”. In: *arXiv preprint arXiv:2112.14887*.

Kung, Pou-Chun, Chieh-Chih Wang, and Wen-Chieh Lin (2021). “A normal distribution transform-based radar odometry designed for scanning and automotive radars”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 14417–14423.

Ng, Yin Zhi et al. (2021). “Continuous-time Radar-inertial Odometry for Automotive Radars”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 323–330.

Retan, Kyle, Frasher Loshaj, and Michael Heizmann (2021). “Radar Odometry on $SE(3)$ With Constant Velocity Motion Prior”. In: *IEEE Robotics and Automation Letters* 6.4, pp. 6386–6393.

Kalman, Rudolph Emil (1960). “A new approach to linear filtering and prediction problems”. In.

Haug, Anton J (2012). *Bayesian estimation and tracking: a practical guide*. John Wiley & Sons.

Sola, Joan (2017). “Quaternion kinematics for the error-state Kalman filter”. In: *arXiv preprint arXiv:1711.02508*.

Madyastha, Venkatesh et al. (2011). “Extended Kalman filter vs. error state Kalman filter for aircraft attitude estimation”. In: *AIAA Guidance, Navigation, and Control Conference*, p. 6615.

Blanke, Mogens et al. (2016). *Diagnosis and fault-tolerant control*. Vol. 3. Springer.

Willersrud, Anders et al. (2015). “Drillstring washout diagnosis using friction estimation and statistical change detection”. In: *IEEE Transactions on Control Systems Technology* 23.5, pp. 1886–1900.

Galeazzi, Roberto, Mogens Blanke, and Niels Kjølstad Poulsen (2012). “Early detection of parametric roll resonance on container ships”. In: *IEEE Transactions on Control Systems Technology* 21.2, pp. 489–503.

Fang, Shaoji, Mogens Blanke, and Bernt J Leira (2015). “Mooring system diagnosis and structural reliability control for position moored vessels”. In: *Control Engineering Practice* 36, pp. 12–26.

Kay, Steven M (1998). “Fundamentals of statistical signal processing. detection theory, volume ii”. In: *Printice Hall PTR* 1545.

Hansen, Soren and Mogens Blanke (2014). “Diagnosis of airspeed measurement faults for unmanned aerial vehicles”. In: *IEEE Transactions on Aerospace and Electronic Systems* 50.1, pp. 224–239.

Technical
University of
Denmark

Ørsteds Plads, Building 348
2800 Kgs. Lyngby
Tlf. 4525 1700

https://www.elektro.dtu.dk/