

Evaluation of Low-Cost INS and Forming a Georeferenced 3D Point Cloud

Peter Eriksson

36657

Master's thesis in Computer Engineering

Academic supervisors: Annamari Soini, Sebastien Lafond, and Dragos Truscan

Advisor: Kai Jämsä

Åbo Akademi University

Faculty of Science and Engineering

Department of Information Technologies

November 2018

Abstract

This Master's thesis demonstrates the possibility of utilizing both inertial measurement units as well as global navigation satellite systems to create a georeferenced point cloud while simultaneously increasing the accuracy of a point cloud created with a LiDAR. To achieve this, an inertial navigation system will be implemented in a modular way to be able to utilize raw data from the inertial measurement units as well as the accurate global positioning data. To achieve a modular system, ROS will be used as a middleware. Once the inertial navigation unit is completed, it will be compared to a commercial off-the-shelf solution by taking a pre-determined path on a bike while both systems are logging the location data. Once the comparison between the two systems has been made, the inertial navigation system will be integrated into a LiDAR scanning device to determine the effects that the accurate location data has on building point clouds. Moreover, since the inertial navigation unit utilizes inertial measurement units, we also want to find out whether tracking the LiDAR system movements can increase the accuracy of the point cloud. The experiments showed that the custom inertial navigation system, though not quite as accurate as the commercial system, was comparable in accuracy when considering the large price difference. However, the point cloud experiment did not show an improvement in accuracy when using the inertial navigation system in the current use case that the LiDAR scanner was to be used for.

Key words: INS, IMU, LiDAR, GNSS, Data fusion

Table of Contents

1. Introduction.....	1
2. Inertial navigation	3
2.1 Inertial Measurement Unit.....	4
2.2 Microelectromechanical Systems	5
2.3 Gyroscope	6
2.4 Accelerometer.....	7
2.5 Magnetometer	8
2.6 Position and orientation	10
2.7 Sensor fusion	11
3. Global Navigation Satellite System	14
3.1 GPS	17
3.2 GLONASS	18
3.3 Galileo.....	19
3.4 INS/GNSS fusion.....	20
3.5 Offset compensation	22
4. Robot Operating System	23
5. LiDAR.....	26
5.1 Point Clouds.....	27
5.2 Simultaneous Localization and Mapping (SLAM).....	29
6. Implementation	31
6.1 INS construction	31
6.2 Design of the INs test platforms	32
6.3 Design of software	33
6.4 Commercial system: Applanix APX-15	36
7. Evaluation	37
7.1 Research question 1: INS vehicle experiment	37
7.1.1 Experiment setup.....	37
7.1.2 Experiment hypothesis	38
7.1.3 Experiment Process	39
7.1.4 Experiment results.....	41
7.1.5 Experiment conclusions and observations	47
7.2 Research question 2: Loop closing improvement.....	48

7.2.1	Experiment setup.....	48
7.2.2	Experiment hypothesis	49
7.2.3	Experiment Process	50
7.2.4	Experiment results	50
7.2.5	Experiment conclusions and observations	53
8.	Summary of work.....	53
9.	Future work	55
	Swedish summary.....	56
	References	62

Foreword

This thesis is about creating a georeferenced point cloud using inertial measurement units and a GPS receiver while simultaneously using the inertial measurement units to increase the accuracy of the point cloud. This thesis was commissioned by Brighthouse Intelligence Oy, who needed a way to make a georeferenced point cloud scan using an inertial navigation system and to compare the system to be created with a commercial INS. This led to discovering the need to reduce measurement error in a point cloud caused when moving the LiDAR scanner around and subsequently discovering the possibility to utilize global positioning data to increase the loop closing accuracy in point clouds.

I express my gratitude to Brighthouse Intelligence Oy for accepting me to work and write my Master's thesis for them, and to all the employees that helped me with different problems. Special thanks to Kai Jämsä for being my supervisor at Brighthouse Intelligence, Matias Tamminen, Benjamin Biström, Samuel Klas, and Sebastian Bröckl.

Thank you to Annamari Soini and Dragos Truscan at Åbo Akademi University for being my supervisors.

List of abbreviations and terms

GNSS – Global Navigation Satellite System

GPS – Global Positioning System

INS – Inertial Navigation System

LiDAR – Light Detection and Ranging

IMU – Inertial Measurement Unit

MEMS – Microelectromechanical System

6-DOF – Six Degrees of Freedom

AHRS – Attitude and Heading Reference System

FOG – Fiber Optic Gyroscope

SQUID – Superconducting Quantum Interference Device

GLONASS – Globalnaya Navigatsionnaya Sputnikovaya Sistema

CA – Course-Acquisition

PDOP – Positional Dilution of Precision

WGS – World Geodetic System

RAIM – Receive Autonomous Integrity Monitor

DGPS – Differential GPS

SBAS – Satellite-Based Augmented System

RTK – Real Time Kinematic

SPS – Standard Positioning System

PPS – Precise Positioning System

FDMA – Frequency-division multiple access

CDMA – Code-division multiple access

NMEA – National Marine Electronics Association

GSA – European GNSS agency

EKF – Extended Kalman Filter

UKF – Unscented Kalman Filter

UT – Unscented Transform

MS AUKF – Master Slave Adaptive Unscented Kalman Filter

FOV – Field of View

LAS – Laser Areal Survey

PLC – Point Cloud Library

SLAM – Simultaneous Localization and Mapping

LOAM – LiDAR Odometry and Mapping

ROS – Robot Operating System

VSWR – Voltage Standard Wave Range

1. Introduction

Navigation is an essential part of traveling from point A to point B without becoming lost on the way, while avoiding obstacles that may hinder the way and making sure that point B is the exact place where we want to arrive at. There have been many different methods of navigation over the centuries that man has used to travel over land and sea, starting with using known places and objects combined with simple measurements of distance and time, to using the sun and stars, and ultimately using advanced satellite technology. Today the most popular way of navigation is using Global Navigation Satellite Systems (GNSS) to receive a current position and other data overlaid on a map. The first and most used GNSS, the Global Positioning System (GPS), was developed by the United States Department of Defense for military and intelligence use but was later released to the public. Other countries have developed their own GNSS solutions over the years, but GPS is the most well-known.

The main problems with using GNSS in applications that demand real-time positional data is that new positional data, using a low-cost GPS receiver, is received at a rate of 1 Hz, and requires that the system has a clear view of the sky so that the receiver can “see” the satellites. If we have a fast-moving vehicle that needs to have an accurate location at all times to avoid obstacles, or if we need to know exactly the route that a vehicle has traveled with near sub-meter, sub-decimeter or even sub-centimeter accuracy, a refresh rate of 1 Hz is not fast enough. To solve these problems, an Inertial Navigation System can be used.

Two research questions will be answered while implementing a custom INS. The first research question is, if a cheaper custom system is comparable to a commercial one when it comes to accuracy. The second research question is, if the use of an INS with a LiDAR can improve the accuracy of the point cloud, when used in a specific application.

The solution will give Brighthouse Intelligence a system that can be easily modified to any future application as well as being fast, inexpensive and easy to repair in case a component stops working. Moreover, being able to use the same system in a multitude of different applications will eliminate the need to find and buy a new system for every

new application. Another added benefit is the possibility to use more than one system in an application due to its small size and low cost, resulting in a redundancy in the application as well as possibly increasing positional accuracy. There will be some downsides to the system. For example, due to the low cost of the components there is a possibility that the components are not as accurate as commercial alternatives. Another downside will be that there will be no technical support in case the device stops working.

The solution will be comprised of three separate smaller systems implemented in the Robot Operating System (ROS), a requirement of Brighthouse Intelligence. The first system will take care of generating the data between GNSS positions, the second system takes care of the GNSS signals, and finally the third system combines the two previous systems to produce accurate location data. The INS and GNSS systems must be able to function independently so that if the GNSS system loses sight of all satellites, the INS must continue to provide location data. If both the INS and GNSS stop working, the system that produces the accurate location data could utilize other implemented odometry systems, but that would be future development.

The main components of the system will be two Inertial Measurement Units (IMUs), a GNSS receiver, and a computer that will combine the sensor data received from the IMUs and GNSS. The outputs of the system will be at least three: the main output which will be an accurate position, raw IMU data, and attitude data will be outputted by the system. These can and will be used in other systems, such as the LiDAR system which will be used to build a georeferenced point cloud of an area. To test the accuracy of the new systems a series of experiments will be performed. A predetermined route will be planned and taken using the new INS/GNSS system, as well as using a different, commercial, INS/GNSS to compare and see if there is a significant difference between the performance of the two systems. In our case, I will be using an Applanix APX-15 positioning system which is an INS/GNSS that currently costs around \$18,000 and is used in a wide variety of applications. If the custom system is as accurate or nearly as accurate as the commercial system, it could be worth designing a more compact solution that could be sold as a new product.

2. Inertial navigation

Inertial navigation is at its core the process of dead reckoning, also known as deduced reckoning [1]. It is most commonly used when navigating at sea by boat because there is no way of knowing one's position, if there is no land in sight that can be used when looking at a map. By dead reckoning we mean that when knowing approximate speed and heading, we can approximately calculate where we are after a certain amount of time, but this is of course only attainable if we know where we started from. These calculations can be verified and corrected by using more accurate but more difficult ways of calculating one's position, for example using the stars at night, or even the sun during the day utilizing a tool called a sextant. Dead reckoning can also be used when navigating when flying, when navigating in the desert or any place where it is not possible to keep track of one's position by looking at reference points on a map.

Inertial navigation uses inertial sensors, accelerometer, gyroscope, and in some cases magnetometer, to dead reckon. The accelerometer measures accelerations, and if we integrate, gives us the speed, and if the speed is integrated it gives us the distance traveled. The gyroscope measures the rotations, and when combined with an accelerometer we can measure to which direction the accelerations occur, which is the heading. The magnetometer will assist the gyroscope with more accurately determining the heading. Thus, when we combine the two or three sensors we have all the necessary data to achieve dead reckoning. Knowing how one has traveled is only one part of navigation, we also need to know where we started from, the starting position, and a map on which to keep track of where one has traveled. This is when the GNSS and similar systems are useful, that is checking and correcting the data received while using the INS, in the same way that the stars and sun were used in the past. [1]

2.1 Inertial Measurement Unit

To estimate the relative acceleration, velocity and position of the vehicle that we have the custom navigation system in, an inertial measurement unit (IMU) will be used. An IMU is a combination of sensors, usually an accelerometer and a gyroscope, but it sometimes even includes a magnetometer. IMUs are most commonly used in aircraft and ships but recently, with the introduction of Microelectromechanical Systems (MEMS), IMUs have significantly reduced both in weight and size. This has resulted in IMUs becoming popular in smaller unmanned vehicles and robots. The output from an IMU will in most cases be raw data from the sensors, and since this data is used to estimate the heading and attitude of the system, the system can be called an Attitude and Heading Reference System (AHRS) or unit. The device can keep a six degree of freedom (6-DOF) estimation of the vehicle orientation and pose, (x, y, z, and roll, pitch, yaw). How an AHRS is calculated will be explained in more detail in the chapter 2.7 about sensor fusion, once a better understanding of the different sensors in the IMU has been achieved. [2]

There are two different methods of utilizing an IMU, either gimbaled or strapdown. The gimbaled method utilizes a gimbal to keep the unit stable on a platform which is aligned with the reference frame at all times. A gimbal has motors in all axes that counter any movement to the reference body. If, for example, a car has a gimbaled platform mounted on the roof which is level with the road and the car starts to drive up a hill, the gimbal will counter the incline by controlling a motor on the affected axes, keeping the platform in the same level position it started in. A strapdown IMU is, like the name implies, strapped down to the system that is being measured. In a case of a car with a strapped-down IMU, it will be rigidly connected to the car frame, for example the roof. The gimbaled method was more common when IMUs were first developed due to the computational requirements of the strap-down method. Today it is more common to use the strapdown method since gimbals are an extra cost and computers can handle the computational requirements of the strapped-down method. [2]

To compare the performance of different IMUs, there are at least four characteristics that can be used. The first is angle random walk. This measurement comes from the gyroscope and is measured by looking at the amount of noise that is present in the

gyroscope data. The second factor is bias repeatability. Bias repeatability is measured over different time periods and is the maximum deviation in the gyroscope. The measurements are performed under ideal conditions in a controlled environment and can produce a short-term repeatability as well as a long-term repeatability. The third factor is position accuracy. The positional accuracy can be tested for both IMUs and algorithms by changing out one or the other or even both. The fourth and final factor listed is scale factor ratio. This is a general measurement of signal amplitude and is not only used for evaluating IMU performance. The scale factor ratio is measured by having a measurement of interest, for example accelerometer acceleration in m/s^2 , and comparing it to the analog output voltage from the sensor. For an IMU with an accelerometer and gyroscope, the measurements are $mV/(m/s^2)$ and $mV/(deg/s)$ respectively. The different tests that will be done on the system to verify the systems' capabilities will be explained in detail in the experimentation chapter 7. [2]

2.2 Microelectromechanical Systems

Microelectromechanical systems (MEMS) are at their core small sensors, actuators, structures and microelectronics that are built on a micro scale, where the components are smaller than a millimeter. Most MEMS components use electrostatic force and electricity to work, though not all MEMS devices use electrical components. Even though 'mechanical' is in the name, not all MEMS devices have moving parts. The fact that MEMS components do not need to have mechanical or electrical parts to be called MEMS can be confusing and thus many use the name microsystems instead, mainly in European countries. [3]

MEMS components are used in a wide variety of different applications, from mobile phones to cars and airplanes, as well as bio- and chemical sensors. Since the components are on a micro scale, they introduce numerous physical phenomena that are observable, such as vibrations, which might be neglected or unobservable on a normal scale. Even compared to macroscale components, MEMS will have a completely different operating principle. There are many other reasons to go down to a micro scale. The most obvious reason is that less materials are needed to manufacture the components, which will also result in a less expensive component. This will also

allow manufacturers to utilize batch processes when fabricating MEMS components. Another reason for utilizing micro components is that one can use several of them on the same area as a single larger component. This feature is something I will utilize, since I will be using two IMUs instead of just one, for possibly increased accuracy and redundancy in the system. Since the MEMS components are small and inexpensive, they can easily be disposed of after being used in environments that are either biohazardous or chemically hazardous. [3]

MEMS components are fabricated using tools and techniques similar to those used when fabricating integrated circuits and semiconductors.

2.3 Gyroscope

The first gyroscope construction was created by Bohnenberger in the year 1817 [2] and Martinus Gerardus van den Bos patented the gyrocompass in 1885. Since then the gyroscope has decreased dramatically in size and increased in accuracy. The main principle behind both gyrocompasses and gyroscopes is the conservation of angular momentum; if there is no external torque applied to a rotating object it will continue to rotate around the same axis and at the same angular speed. Gyroscopes are good for applications where the local change in orientation is needed, and not an absolute orientation due to the global motion of the earth's rotation. The two most popular gyroscope systems today are fiber optic gyroscopes (FOGs) and MEMS gyroscopes. Since I will be using MEMS gyroscopes in the custom system being built, I will focus on how they work and only briefly explain the difference between a FOG and MEMS gyroscopes. [2]

FOG systems utilize a glass fiber that is looped multiple times, and two laser light beams in opposite directions to detect the time delay that is introduced when the system rotates. The effect that makes this possible is the Sagnac Effect which was discovered by Georges Sagnac. The reason for not using a FOG in the custom system being built is the price of the units and their weight and size. FOGs are more expensive and much larger than MEMS-based gyroscopes.

MEMS-based gyroscopes sense rotations by utilizing vibrating mechanical elements, based on the Coriolis acceleration. This acceleration is given by $a = 2v * \Omega$ where v

is the local velocity and Ω is the rotational rate. The vibrating mechanical elements are made of silicone, but in the past, they were made of quartz crystals. One example of how a MEMS gyroscope can be implemented is the tuning fork gyroscope. As the name implies, the structure of the gyroscope looks like a tuning fork. The rotational force is measured when the tines of the fork vibrate out of the plane. This is caused by the Coriolis forces stemming from the tuning fork's vibrations. A single tuning fork gyroscope can only measure the rotation of a single axis, thus to measure three dimensional rotations, three tuning forks must be used.

The tuning fork gyroscope works in the following way to begin with. The structure will be in constant vibration at a natural frequency, since the Coriolis force is being used and it only works on moving bodies. The motion that the proof masses are vibrating in when the device is at rest is called drive mode. This motion can be described as a side to side, horizontal motion in the x-axis. Once the device starts to rotate, the horizontal motion will change to an up-and-down, vertical motion caused by the Coriolis force, and is called sense mode. To measure the sense mode movement, metal plates are placed above the proof masses to form a capacitor. When the proof mass moves in the drive mode, the capacitance will remain constant, while in the sense mode it will change. [2]

The key problem with all gyroscopes is that they drift, and the drift error accumulates over time, causing corrupt measurements in the long run. Most manufacturers measure and state how much the gyroscope drifts in the supplied datasheet that comes with the gyroscope, this measurement is in degrees per hour.

2.4 Accelerometer

Accelerometers are used to measure external forces, including gravitational force, acting on a vehicle or anything the sensors are placed on. Like the gyroscope, there are many different mechanisms that can be used to measure acceleration and output the data to a computer. The mechanical way of measuring acceleration is to use a spring-mass-damper system and a way to monitor the system. When an external force is applied to the system, the mass will move, and this will cause the spring to also move and thus be displaced. The spring displacement can be measured, making it possible

to calculate the balance between external forces and internal forces. Unfortunately, the large mechanical spring-mass-damper accelerometer has its drawbacks. The system is sensitive to vibrations, and since the system relies on the idea of an ideal spring, which does not exist, the applicability of the mechanical accelerometer is limited. A better solution is to use a piezoelectric accelerometer.

A piezoelectric accelerometer uses a piezo crystal which generates a voltage across itself when stressed, and this voltage can be measured. The accelerometer is constructed by placing a small mass on top of a piezo crystal, so it is only supported by the crystal, and on the opposite end there is a small spring that holds the mass in place. When an external force is applied, the mass will press down on the crystal, thus generating a small voltage that can be measured. This is the reason why gravity is always measured by a piezo crystal accelerometer; there will be a constant voltage over the crystal when the gravitational force acts on the small mass on the crystal. [2]

In MEMS accelerometers the acceleration can also be measured using capacitive sensing. This method uses almost the same idea as the mass-spring-dampener, but instead of measuring displacement of mass, a capacitance is measured. [4]

The proof mass is a movable structure suspended between springs with plates which will, together with the fixed outer plates, result in a capacitor. The base that the springs are attached to is the reference frame, i.e. the frame that will be affected by the acceleration force to be measured. The acceleration equation comes from Newton's second law, not taking into account the air friction due to the structure's small size.

2.5 Magnetometer

The compass has been a vital part of navigation since it was discovered, and is utilized to achieve heading when used in an INS. There are many different technologies that can be utilized when it comes to detecting magnetic fields. The most sensitive according to A. L. Herrera-May et al. [5] is the Superconducting Quantum Interference Device (SQUID), which utilizes effects only observable when a superconductor is used. The SQUID is very expensive and requires a lot of special equipment such as liquid helium to operate, therefore the SQUID is not widely used. For MEMS technology, the Hall effect sensors and resonant magnetic field sensors are the most

common. Emphasis will be placed on the resonant magnetic field sensors since the technology has its advantages over Hall effect sensors. [5]

Resonant magnetic field sensors utilize the Lorentz force to detect magnetic fields, and there are at least three different methods to measure the displacement caused by the Lorentz force acting on a resonant structure. The methods are optical, piezo resistive, and capacitive, which also have different methods of manufacturing. In general, the exact technology used to measure the ambient magnetic field in different magnetometers is not listed in their datasheet and is most likely a manufacturing secret.

The magnetometer is not susceptible to noise in its readings, but if there are any power cables or motors near the sensor, they will cause a lot of magnetic disturbance. When installing a magnetometer, it is therefore best to avoid having power cables near the sensor and to also avoid installing the sensor on large metal surfaces. Magnetometers also need to be calibrated according to where in the world the sensor is, due to the fluctuation in earth's magnetic field as well as distortions caused by objects near the sensor. These distortions are called hard iron and soft iron distortions respectively. Hard iron distortion in general causes more error in the measurements than soft iron distortions. The magnetometer should therefore be re-calibrated before each use of the sensor, but this usually requires the sensor to be rotated in all directions which in some cases is not possible to do.

When using a magnetometer, once it has been calibrated it will output 0 when facing either north, west, south or east. This depends on the manufacturer of the IMU/magnetometer. In our case the magnetometer outputted a value of 0 when facing east. This means that we always know where east is, and we can from this determine the heading of the system, that is in which direction our device is traveling. It is necessary to know in which direction the magnetometer will show a value of 0, to be able to compute the correct heading.

Comparing the different sensing options when it comes to resonant sensors, they all have advantages and disadvantages. Depending on the manufacturer and what end use the magnetometer will be used for, the appropriate options will be chosen. [5]

2.6 Position and orientation

To be able to understand how a vehicle moves and its position in space or on earth, there must be a way to represent position and orientation of an object. For representing position there are not many formats to choose from, usually a 3×1 vector is used [10].

In the position vector we have $({}^j p_i^x, {}^j p_i^y, {}^j p_i^z)$, which are the x, y, z coordinates of a frame i relative to the coordinate frame j . For a navigation system, a (latitude, longitude, altitude) coordinate frame is used.

As for orientation, there is more than one way to represent orientation and no single representation is better than the other, they only solve different problems. The most common representations of orientation and rotation when it comes to robotics are rotation matrices, Euler angles, fixed angles, angle-axis and quaternions. Focus will be placed on the quaternion representation since it will be the choice of representing orientation in the system that is being developed. Unlike Euler angles and fixed angles, quaternions do not suffer from singularities. A singularity is a point where an object, in this case an orientation, is not defined or does not behave well. The form of a quaternion is the following:

$$\epsilon = \epsilon_0 + \epsilon_1 i + \epsilon_2 j + \epsilon_3 k$$

where i, j, k are operators and ϵ_{0-3} are scalars which are also called Euler parameters. The following rules are satisfied according to the definition of the operators:

In the process of adding two quaternions, the operators act as separators and the components are added separately. Quaternions have different null elements for addition and multiplication, for addition $\mathbf{0} = 0 + 0i + 0j + 0k$ and for multiplication $\mathbf{1} = 1 + 0i + 0j + 0k$. Addition and multiplication also have different properties, where addition sums are associative, commutative and distributive, multiplication products of quaternions are only associative and distributive but not commutative. [10]

There is also the possibility to first calculate the quaternion orientation of each IMU and then calculate the average of the two quaternions based on an equation from [11]. This method could be implemented in the custom system being built and compared to the averaging of raw IMU data in the future, but that falls out of the scope of this thesis.

2.7 Sensor fusion

In general, when a robust and complete description of a process or environment is needed, combining data from multiple sensor data can be used. This is known as sensor fusion or data fusion. When using an IMU to calculate the Attitude and Heading Reference System (AHRS) for a vehicle, the sensor measurements must be combined. In the case of an IMU, data from the gyroscope, accelerometer and magnetometer, if available, are fused. When fusing only data from an accelerometer and gyroscope, the accelerometer is used for acceleration estimation a and from the gyroscope, the angular velocity ω , is integrated to achieve an orientation estimation θ . Next, the gravitation is removed by transforming the data to the orientation of the vehicle which was obtained from integrating the gyroscope data. Once the gravitation is known and removed, the acceleration of the vehicle is obtained and from the acceleration, through integration, velocity is obtained and if velocity is integrated the distance traveled, position, is obtained. Position and heading are the necessary components for dead-reckoning, they are obtained from the sensor fusion. [2]

The problem with sensor fusion when using an IMU is the fact that the sensors, accelerometer and gyroscope, are extremely sensitive to errors. Since the gyroscope drifts over time, the orientation of the vehicle will not be true, which leads to errors when removing the gravitation from the acceleration data. This leads to errors in both the velocity and position data, since the accelerometer data is integrated to obtain the data. To correct the error, a correcting measurement must be implemented, which in most cases for an INS is a GNSS solution. [2]

In the custom solution two IMUs will be used to increase accuracy. This has already been demonstrated in [6,7] to be possible. There are many different algorithms or filters that can be used to solve IMU sensor fusion. One widely used filter is the Kalman filter, based on Bayes' rule, which utilizes prior and observation information and combines them to an estimated value. For the custom solution, different orientation filters will be tested to see which one works best with the chosen IMUs.

A simple filter when fusing IMU data is the complementary filter, which combines a low-pass filter and high-pass filter on the different sensors. The high-pass filter filters the gyroscope data while accelerometer and magnetometer data will pass through the

low-pass filter. The complementary filter can also be used to combine GNSS data to the IMU data. However, for this there are better ready implemented solutions. One implementation of the complementary filter can be found here [8] [40].

Another filter which will be tested, slightly more complicated than the complementary filter, is the Madgwick algorithm [9] which is specifically tuned to fuse IMU data to estimate the orientation. The Madgwick filter comes in two different implementations, one that only fuses the accelerometer and gyroscope data and one that fuses accelerometer, gyroscope and magnetometer data. We will use the one that fuses the data from all three sensors in our IMU if the tests show that using the magnetometer yields more accurate results than not using it.

The Madgwick algorithm fuses orientation from angular rate and the estimated orientation based on a previous estimation of orientation. First, the quaternion representation, explained above, needs to be defined and this can be combined with the gyroscope measurements of rotation to calculate the rate of change of orientation. Using the quaternion derivative, the orientation of the earth frame relative to the sensor frame can be described. Knowing the initial condition of the system, the quaternion derivative can be integrated to obtain the orientation at time t .

Since gyroscopes are known for drifting over time, a gyroscope bias drift compensation is implemented in the filter. The compensation takes the gyroscope bias and subtracts it from gyroscope measurements. The gyroscope bias is calculated from the integral of the angular error in each axis weighted by a gain.

Next the accelerometer and magnetometer will be used. The accelerometer will measure acceleration, gravity and its direction while the magnetometer will measure the earth's magnetic field and its direction, as well as the magnetic flux and distortion around the sensor. When the system is started, the filter will assume that the only measured data are acceleration and the earth's magnetic field. For optimizing, the filter uses a gradient descent algorithm. The equation for calculating the estimated orientation can be found in more detail in [9] along with all the other necessary equations for the IMU orientation filter.

The orientation estimation is based on the previous orientation estimation. The accelerometer, magnetometer and gyroscope orientation equations can then be fused

by combining their separate equations and this filter uses a weight variable γ_t which can be tuned to increase the accuracy of the filter.

To give a better picture of how the different parts of the equations are utilized and in which parts of the system they are used. Figure 1 provides a block diagram representation which can be used to easier understand the filter. The complete implementation can be found in [9].

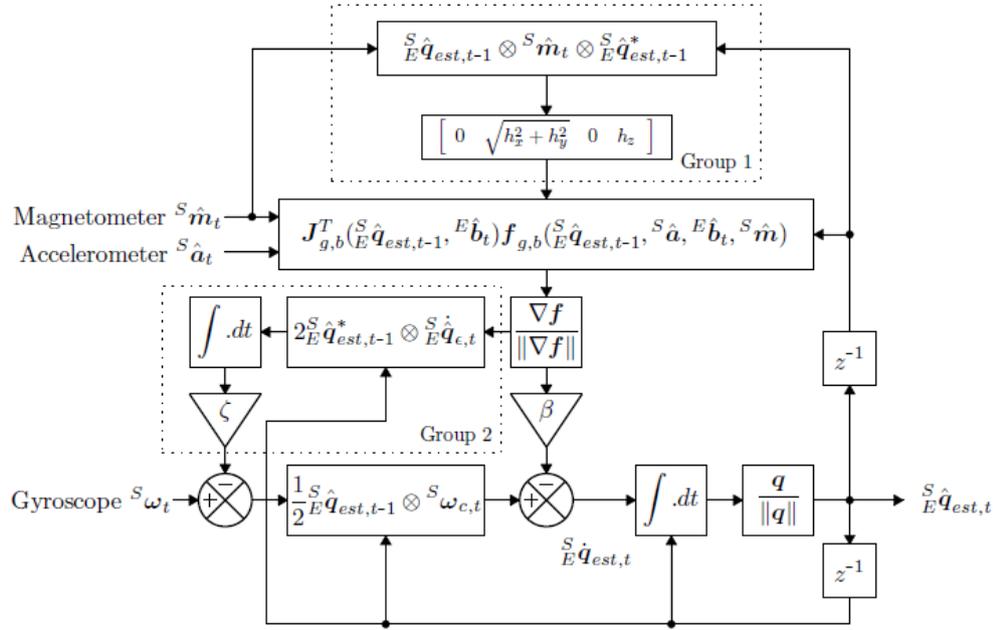


Figure 1: The complete Madgwick orientation filter. Group 1 are the magnetic distortion calculations and group 2 are the compensation calculations for gyroscope drift. [9]

3. Global Navigation Satellite System

The Global Navigation Satellite System (GNSS) is a mechanism for three-dimensional location estimation, output as absolute coordinates. The most commonly used instance of GNSS is the global positioning system (GPS) which also provides accurate date and time information and can be used anywhere on the planet as long as there is a clear view of the sky. GPS, built and run by the United States of America, is not the only GNSS available, it was just the first one to be developed and therefore the first system to be widely used. Russia, China, the European Union, India and Japan all have their own GNSS programs either already up and running or under development. When one buys a GNSS receiver, the receiver is most likely going to use GPS positional data, but it is becoming more common for receivers to be able to receive data from more than one system, commonly the Russian GLONASS (*Globalnaya Navigatsionnaya Sputnikovaya Sistema*).

The way that a GNSS system works is that the satellites broadcast their own position as well as the time that the message was sent to earth and received by the GNSS receiver. The data sent out is commonly known as Coarse-Acquisition (CA) code. Once the message is received by the receiver, it uses trilateration to determine its position. Figure 2 shows how trilateration requires three satellites to be used. A fourth satellite is required if altitude measurements are needed. The way that trilateration works is, when the receiver has one satellite lock, there is an approximate circular line where the receiver could be. Adding another satellite there will be two circles and now two possible positions the receiver could be, where the circles intersect. Finally, once the third satellite locks, its circle will intersect at one place with the two other circles and form an accurate location where the receiver is. Trilateration is the general way all GNSSs work, more

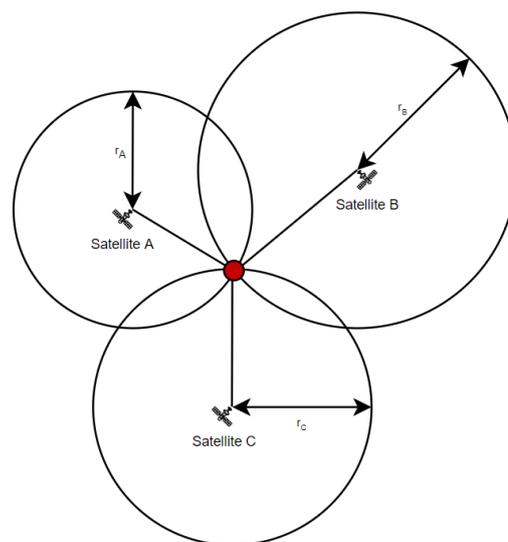


Figure 2: Trilateration in 2-D. Knowing the delay between transmission and receiving a location message, we obtain distance/radius r . [2]

location where the receiver is. Trilateration is the general way all GNSSs work, more

detailed explanations on the methods the different GNSSs use to increase accuracy are to be found under their respective sections. [2]

In the following, the algorithm for GPS is explained in detail since that is the one used in this work and it does not differ too much from the other GNSSs. First the receiver will select four satellites based on Positional Dilution of Precision (PDOP), which means that it will choose satellites that are furthest apart from each other. This will improve the accuracy. A new PDOP calculation is performed, updating the last one, whenever a satellite in use sets, or every five minutes. Next, the pseudo range to each satellite is measured, which is the distance between each satellite and the receiver. This is performed every half a second. The *pseudo* comes from the inaccuracy and possible errors in the distance measurements. There are two timers used when calculating the pseudo range: the transmission time received from the highly accurate clock on the satellite and the reception time which is taken from the receiver's own clock. Once the pseudo ranges have been calculated, the position of the satellites, ephemeris, must be determined and from that we can calculate the satellite earth-centered, earth-fixed coordinates. This coordinate system has its origin at the center of the earth. Next, a new pseudo range must be calculated that takes into account the earth's rotation. Comparing the two different pseudo ranges, subtracting one from the other, results in the range residuals. After calculating the range residuals, we must estimate the position solution geometry matrix which is used to determine the overall system solution. The matrix includes a collection of row vectors with a vector x, y, z , direction cosines, expressed with the coordinate time of the satellite, between the user and the satellite using the World Geodetic System (WGS) 84 standard [41]. When all of this data has been calculated, the final location of the receiver can be calculated. This algorithm is run every second, or according to the GPS receivers' specifications. [2]

With the above-mentioned algorithm, the accuracy that is possible using the non-encrypted GPS signal is 20-25 meters horizontally and 43 meters vertically, which is not very accurate. Using the encrypted signal improves the accuracy slightly, typically to 7-10 meters horizontally and 27.7 meters vertically, which is better but for some applications, such as autonomous navigation or geolocating a point cloud, it is not accurate enough. Thankfully, there are many ways to improve the accuracy, both on the receiver chip and using other systems to complement the basic satellite position

algorithm. On the receiver chip, the following methods can be used to increase the overall accuracy of the system: multiple satellite solutions, range residual smoothing, Kalman filtering, Receiver Autonomous Integrity Monitor (RAIM) and velocity aiding. Systems that complement, or enhance, the GPS are differential GPS (DGPS), Satellite-Based Augmented System (SBAS) and Real-Time Kinematics (RTK). Using these the accuracy can be increased from the original 20-25 meters to 1-2 meters, and all the way to 1 centimeter when using RTK. Depending on the price to implement RTK, the custom system might use it in the future. DGPS, SBAS and RTK all work in a similar way, utilizing one or several base stations with a known accurate location that send out correcting data to the GNSS receiver in an unknown location over radio frequencies between 285 and 325 kHz. RTK uses a similar method utilizing a single base station which can send correcting data over a network instead of radio. DGPS works over a range of several hundred kilometers while RTK has a range of about 20 kilometers around the base station. RTK solutions are, however, much more expensive than DGPS solutions since one needs to buy the base station and a compatible GNSS receiver. [2]

All GNSSs have a public and an encrypted broadcast channel. The public broadcast is less accurate compared to the encrypted broadcast which is used either for commercial applications or military applications. The two different channels were most likely created to counter the use of GNSS in terrorism or similar cases and to be able to monetize the usage of the system. [2]

The key component that determines the accuracy of the positioning data is the clock in the satellites. It is easier and less expensive to install a highly accurate clock in a satellite instead of in the receiver. The clock is used to calculate the distance from the satellite to the receiver and the more accurate the clock is, the more accurate the distance calculation will be. All the satellites have very accurate atomic clocks installed, even multiple ones for redundancy, to achieve the best possible time accuracy. [2]

GNSS suffers from some limitations that will reduce the accuracy of the location data received or completely stop location tracking. The ionosphere that the satellite signals travel through will distort the signal somewhat and this will result in a slight inaccuracy. To eliminate the inaccuracy, it is possible to either use a model for how

the ionosphere distorts the signal or, since the satellites send the data on multiple frequencies, the two frequencies can be used to estimate and eliminate the inaccuracy. Another signal-related limitation is that the satellites use frequencies in the microwave band, 100 MHz – 300 GHz, to send data to the earth. These frequencies can travel through glass and plastics but are reflected by many materials and absorbed by some materials, for example wood, water and leaves absorb signals. This results in problems when trying to navigate in densely wooded forests, urban environments, etc. since the trees would absorb signals, and buildings either block signals, or the signals reflect off a building and give the receiver a false signal travel time. There are ways of overcoming these errors and these will be discussed later in chapter 3.4. [2]

3.1 GPS

Since GPS was the first GNSS to be developed, it is in nearly every GNSS receiver module and will have the best coverage around the world. GPS also has the greatest number of active satellites currently in orbit, 31 to be exact. Thus, it is nearly impossible not to utilize GPS in a localization system unless the module is configured by the user not to utilize the GPS satellites. The other GNSSs will increase the accuracy of the GPS localization depending on where in the world the system is being used, and on the slim chance of the US government turning the GPS off, the custom system being built will have the possibility to utilize other satellites. [12]

Based on the NAVSTAR system, GPS was developed and maintained by the Air Force Space Command for military usage and therefore the United States government has total control over the GPS configurations. As mentioned in the previous chapter, all GNSSs have a public and a commercial/military encrypted channel. The GPS public channel is referred to as the Standard Positioning System (SPS) which has lower accuracy, and the encrypted one is referred to as the Precise Positioning System (PPS). The reason for the reduced accuracy of the SPS was mostly strategic and was made possible by introducing pseudo noise in the signal. The noise was eliminated on May 2, 2000 and newer satellites are not equipped with noise generators. The system, however, does have limitations built-in to the chips themselves to eliminate the possibility to use them in missiles. When the receiver travels over the speed of 1,900

km/h and/or reaches an altitude of 18,000 meters it will automatically turn off or disable itself, as this would mean that the receiver is in a ballistic missile or a similar weapon. [2]

In total there are 31 operational satellites currently in use, more than the designed 24 in the system. This is to reach a goal of 95% availability of 24 satellites at all times. The 24 satellites are positioned so that at any place on the globe there should be at least four satellites that can provide location data to the user. The additional seven satellites are for situations where any of the base 24 satellites are under maintenance and the seven others will take their place to maintain global coverage. In 2011 the number of satellites in the constellation were expanded to 27 satellites to improve coverage. The Air Force had to reposition six satellites in total to be able to fit the three additional satellites into the constellation. [12]

3.2 GLONASS

As mentioned previously, GLONASS is the Russian GNSS version of the US GPS which has only recently started to become more commonly utilized in civilian navigation systems. One drawback that GLONASS had until recently, was the fact that few GNSS modules had been developed for the civilian and commercial market. The reason for this might have been that GLONASS uses Frequency-division multiple access (FDMA) as the channel access method while GPS uses Code-division multiple access (CDMA), requiring different hardware. Fortunately, GNSS receiver manufacturers are now releasing modules that can use GLONASS as well as GPS at the same time. Since the GLONASS system was developed for the Russian government and market, the coverage is better over the Russian territories and neighboring countries such as Finland. Thus, utilizing GLONASS for the system is useful since it does not increase the cost of the system when buying the GNSS receiver.

The first deployment of GLONASS, originally called “Cicada” or Tsikada system, was commissioned in 1979, by The Russian Federation, with 4 satellites. The system sent a continuous signal on 150 MHz and 400 MHz, providing navigation signal transmissions. Since there were only four satellites in orbit at an altitude of 1,000 km with an inclination of 83°, the users were only able to acquire a satellite lock every

hour and a half or two hours, and it took around 5-6 minutes to acquire a position. Emphasis was placed on calculating the position of the satellites and predicting the orbit. The “Cicada” system was later repurposed and used as a search and rescue service, named “Cospas”, which could detect distress beacons that were then relayed to ground stations. The “Cicada” system was decommissioned 2008 due to the new GLONASS satellites and the fact that the old system was not able to handle the large number of users that required positional data. Today GLONASS is a fully functional GNSS with a total of 24 active satellites, comparable to the US GPS, and has begun to utilize CDMA for its channel access method while still keeping the older FDMA method too. [13]

3.3 Galileo

The Galileo GNSS is a joint program between the countries in the European Union and was first mentioned in 1999 in the Communication of the Commission [37]. The European Space Agency oversees the development while the European Commission will manage the system once it is fully functional [14]. Similar to how GLONASS was not utilized by GNSS modules until recently, Galileo suffers from the same problem since it is such a new system. Fortunately, there are modules that are relatively low-cost that can be utilized and support the three GNSSs mentioned above. Tests done by the European GNSSs Agency (GSA) from 2014 [15] show an increased accuracy when navigating using GPS and Galileo simultaneously. This is the reason why the custom system will utilize all three GNSSs while at the same time adding redundancy in case of the slim possibility that one of the GNSSs is switched off.

There are two phases in the development of the Galileo system. The first phase was to test the hardware in the satellites and to create a functional four satellite GNSS. The second phase of the Galileo system is the “Fully Operational Capacity” phase where all parts of the planned Galileo system are up and running. The full system will include thirty satellites in orbit, sixteen sensor stations, two control stations, five mission uplink stations, five telemetries, tracking and command stations, and four service facilities. The satellites have both navigation payload as well as a search and rescue transponder. [15]

As mentioned before, the time accuracy of the satellites is the key to increasing location accuracy. Galileo satellites use two different atomic clocks, a passive hydrogen master clock and a rubidium clock. Both clocks are extremely accurate, the passive hydrogen master clock can measure time to within 0.45 nanoseconds over 12 hours and the rubidium clocks can measure to within 1.8 nanoseconds over 12 hours. The only downside with the clocks is that they will drift over time if they are left on. To eliminate the drift, the clocks are synchronized with clocks that are on the earth in different ground stations. The achieved time accuracy of the system is 28 billionths of a second, or 28 nanoseconds. [16]

3.4 INS/GNSS fusion

As mentioned in the section about IMU sensor fusion, a GNSS can be used to complement an IMU to correct its drift in data. The problems that the IMU solves for the GNSS, at least for short periods of time, are orientation, continuous position estimation, and positional data when GNSS fix is not obtainable. There are several different filters that can be used to fuse the IMU data with the GNSS data. The complementary filter, for example, is one of the least difficult to implement and does not require any difficult equations but does not meet the precision requirements of the custom system. One of the most popular filters that is used for sensor fusion is the Kalman filter, which is a recursive linear estimator. The Kalman filter is based on Bayes' rule of joint probability where we have the state and the observed state probabilities. The Bayes' rule is the base of many different state estimation filters. When we have multiple sensors, the Bayes' rule will change to a product of the probability of the observed state. To see all the required functions to implement the Kalman filter, see the chapter on multisensor data fusion in the *Springer Handbook of Robotics* [17].

Like the Bayes' filter, the Kalman filter is divided into a time update or prediction step and an observation step. For both the steps the Kalman filter has different explicit statistical models of how they change and evolve over time. The Kalman filter is the base for how I will fuse the INS and GNSS data.

There are many different implementations of the Kalman filter, depending on the requirements of the system and the data that are being fused. Because the regular Kalman filter only works with linear functions and when fusing certain data, such as INS/GNSS data, I need a filter that can better handle non-linear functions. These non-linear functions are obtained from the vehicle dynamics, when calculating how the vehicle moves and estimating where it will be next. The Extended Kalman Filter (EKF) is one implementation that can be used for non-linear functions and is typically used when combining IMU and GNSS data. Another non-linear filter that can be used is the Unscented Kalman Filter (UKF) which is an EKF that uses a number of sigma points instead of a single point as in the EKF when linearizing the non-linear function, known as the Unscented Transform (UT). The UT has been proposed to improve the EKF in [18, 19]. For the custom system, I will be implementing a Master Slave Adaptive Unscented Kalman Filter (MS AUKF) by J. Han, Q. Song and Y. He [20]. The MS AUKF uses two parallel UKF filters, one to determine the state, the master, and one to determine the noise covariance of the master filter, the slave. The adaptive part of the filter uses two filters to adapt faster to changes. This means that one filter takes care of the data fusion part and the other filter will use data from the first filter, basically the error in the system, and filter the error which is then used in the first filter in the next iteration. There are other ways of implementing an adaptive filter than the MS AUKF way, but I decided that this implementation was relatively simple to implement and met the requirements of the system based on the test conducted by J. Han, Q. Song and Y. He [20]. Figure 3 gives an overview of the entire MS AUKF structure.

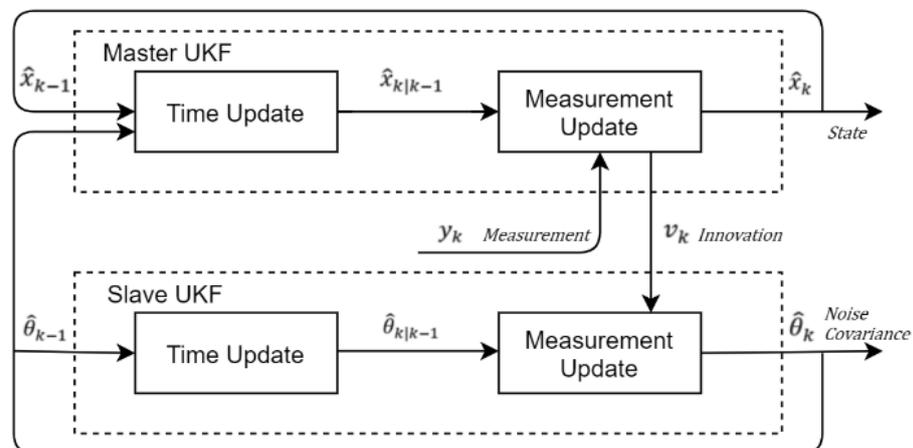


Figure 3: General overview of the MS AUKF structure [20]

To give a general idea of the filter, the following variables and data are needed when implementing the filter. Starting with the dynamics of the covariance matrix, if it is known we can use the dynamics function with the previous covariance matrix and add Gaussian white noise with zero mean to it. Since we do not know the dynamics of the noise in our case, we will use a noise-actuated model where the covariance matrix at time k will be the previous, $k-1$, covariance matrix and add Gaussian white noise with zero mean to it.

From the master UKF we will take the innovation and use it in the slave UKF for the observation model. The innovation can be compared to the GNSS update, as the noise covariance is updated to a better, more accurate, value.

The slave filter does not necessarily have to be implemented using a UKF filter since the equation that the noise covariance implements is linear, a normal KF could be used instead. This would reduce the complexity of the filter as well as reducing the computational power needed for the filter.

3.5 Offset compensation

Something that also needs to be considered when placing the IMU and GNSS antennae is the distance between them. When they are at different locations the IMU will not track the same position of the vehicle that the GNSS antenna is tracking. The offset of the IMU compared to the GNSS antenna can be compensated for by utilizing the gyroscope and measuring the offset. One way to measure the offset acceleration can be found from NASA [42] and a simplified version from [21]. In the offset, acceleration, pitch rate, yaw rate and roll rates are used.

The offset compensation can also be used when the IMU data must be offset to the center of mass of a vehicle, when the pose is important. The ideal solution would be that the IMU and GNSS antennae are placed in the center of mass of the vehicle, but unfortunately this is not always possible. There are also ready implemented solutions that are easier to implement in the custom solution, and better validated ones which work in a similar way. A ready implemented solution will be discussed in the implementation section, chapter 6.

4. Robot Operating System

The Robot Operating System (ROS) had its beginning at Stanford University in the mid-2000's where they started prototyping different frameworks that could be used when developing and creating software for robots. The framework development was boosted in 2007 by a business incubator, Willow Garage, which provided resources to further develop ROS to a well-tested implementation. The first ROS distribution was released in 2010, named Box Turtle. Today ROS is on its twelfth release named Melodic Morenia for Ubuntu 18.04. ROS is widely used by many individuals and different business sectors, with a wide variety of use cases, from creating robots at home to industrial automation systems. [30]

One of the main reasons for why ROS has been so widely used is the collaboration efforts between different institutions, laboratories and individuals. Institutions and laboratories can focus on and contribute to what they have the most knowledge of, for example sensor fusion or mapping, while individuals can implement and test their own versions of the implementations released by the laboratories, institutions and sensor manufacturers. This is one of many reasons why ROS will be used to implement the INS, there are many ready software packages that can be used, eliminating the need to write all code from scratch. Moreover, since the code is open source it can be modified and improved if other features are needed, or there might be a bug in the code that someone else finds and is able to fix. This will also make it easier to incorporate the INS in other projects being developed. More about the different packages that are used will be explained in the implementation section. [31]

The ROS framework can be compared to an operating system, taking care of and providing services that are usually provided by the operating system, such as sending messages between processes, controlling low-level devices and defining hardware abstractions to name a few. ROS can also be used in conjunction with other robot software frameworks and is implemented in several different software languages. Even though it is called Robot Operating System, this does not mean that it can only be used for robots. ROS can be used for anything where different sensors need to communicate with each other and be gathered to be used for, for example, navigating.

ROS needs a main process, called the ROS master, to start and allow the distributed processes to communicate with each other. The distributed processes in ROS are called nodes. The master works like a DNS server in a network, it provides information about the different nodes and messages available in ROS. The nodes only need the master to be able to know where to send messages, the master does not take care of the message passing between them, the nodes communicate directly with each other. The master does not, however, need to be on the same device as the nodes, as it can communicate over a network with different nodes. The nodes only need to know the IP of the computer that the master is located on. However, ROS does not have the possibility, built in, to control several robots at once, and generally one ROS master is one robot.

Nodes can be launched individually, or many at the same time, making ROS very modular. This modularity can be used to create different tailor-made solutions that utilize different sensors, depending on system specifications. There is also no limit to how many nodes can be running at the same time, other than system limitations where nodes start to run slow due to high processor utilization. The modularity of ROS also makes it possible to have one node sending sensor data to many different nodes which utilize the received data in different ways. For example, when using an IMU for calculating steps taken and tracking if someone is sitting or standing, one node can calculate the orientation using the IMU data and another node can use the raw accelerometer data to calculate steps taken.

The messages that are sent within ROS are sent over TCPROS which is the same as regular TCP over a network, but it uses the local hosts network instead of sending the messages out over the network. The messages can either be a standard message pre-defined in ROS, or one can define custom messages that only send the necessary information for a specific application. One example of a standard message is an IMU message, which sends linear acceleration, angular velocity and quaternion orientation in a single message to any node that has subscribed to the message. Since there are standard messages, receiving data from other ready packages, there is no need to write custom parsers as the application only needs to subscribe to the message topic. If there is a need to match two messages sent at the same time by two different nodes, ROS has a master time stamp which is the same for all nodes and each message has a header with the timestamp when it was sent. Unfortunately, ROS is not a real-time system,

which makes it difficult to precisely match timestamps in some cases. Fortunately, there are synchronization tools that can be utilized to either only use messages with the same timestamp, or an approximate algorithm can be used to accept messages with the closest timestamps. Even though ROS itself is not a real-time system it can be used in a real-time system, for example sending messages to other nodes. [32]

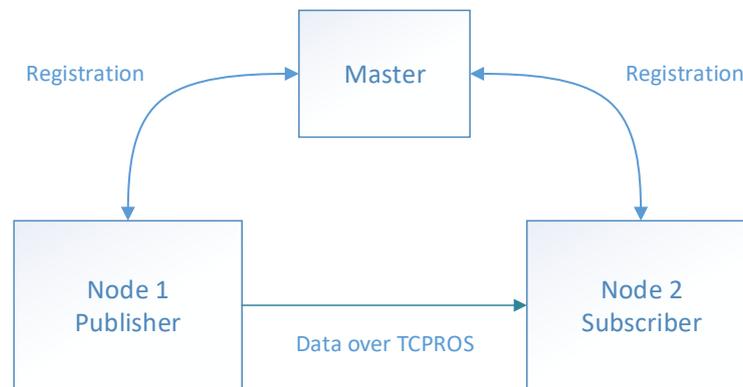


Figure 4 ROS communication example.

In Figure 4, a basic example of what a ROS environment with a master and two nodes could look like. Node 1 will register as a publisher and will tell the master which messages it will publish. The subscriber will register as a subscriber and tell the master which messages it would like to subscribe to. Once the two nodes know about each other, they can start to send and receive data. A node can be both a subscriber and a publisher, for example the custom system will have two IMU nodes that publish to a single combining node that publishes the combined data to another node.

With this straightforward way of sending messages between processes and the modularity of ROS, additional sensors can be easily added to the custom INS in the future by subscribing to the sensor topic being sent by the new sensor and implementing the sensor data fusion in the filter. There might even be a ready implemented package that can be used with the new sensor, making a new sensor integration easy.

5. LiDAR

LiDAR is an abbreviation of *Light Detection and Ranging* which is a method of ranging using light pulses and are used to create point clouds, needed to answer the second research question. The method used to calculate ranges is called *direct time of flight* which uses a source of light, lasers in the non-visible spectrum, and a high-speed chronometer to measure the travel time of the light. The distance is given by the following equation:

$$2d = ct$$

where t is the time of flight, c is the speed of light, dependent on the medium that the light passes through, and d is the distance to the surface that reflects the light. LiDAR accuracy is highly dependent on the timing accuracy, calculating the time from when a laser pulse is sent out to when the reflection is received. To make sure the correct reflection is timed, the peak of the pulsed light is measured, and the time is noted when the reflected pulse is at its peak. This becomes more difficult as the range to the scanned object increases, since the reflected pulses will become weaker as the range increases. The random errors in received data can be reduced by taking multiple readings and averaging them to a single value. [22]

There are many different types of LiDAR devices, from ones that only use a single laser beam to devices that use over 64 laser beams arranged vertically, known as Multiple-Beam scanning LiDARs. LiDAR systems are also available in sweeping configurations where the beams are swept over the area being scanned to obtain more information in a dense point cloud representation. The sweeping motion is achieved by using mirrors and stepper motors rotating the mirrors, since it is easier to move a light and durable mirror instead of a heavy and delicate laser emitter and detector. However, there are LiDAR systems that rotate the emitter and detector. For systems requiring spatial awareness, the more beams the LiDAR has the more information of what objects are around the LiDAR is available.

When multiple beams are pulsing at high rotation speeds and at high rates, the problem of which pulse is received arises. This uncertainty can be used to calculate a maximum distance that is used to know if the measurement is within an interval, the ambiguity interval. The distance can be calculated with the following equation:

$$1/2c\delta t$$

where δt is the time between pulses and c is the speed of light. For example, an ambiguity interval of 1500 meters is obtained if the pulses are repeated at a rate of 100 kHz. This is the maximum theoretical range of the LiDAR but, in practice, it is lower due to processing time. If the pulse is received after another pulse was sent out, the range can be calculated with the following equation:

$$z \bmod 1/2 c\delta t$$

where z is the calculated true range. To obtain the distance, an unwinding algorithm can be used as long as the pulse starts within the interval and the distance values change slowly. [22]

Most LiDARs used in robotics solutions have a typical range from 10-100 m and an accuracy of 5-10 mm. If they have further range and accuracy the price increases rapidly. An example of a LiDAR used in the autonomous car industry and environmental modeling to 3D images, among other things, is the Velodyne VLP-32C “Ultra Puck” [23]. The VLP-32C has 32 beams or channels, a range of up to 200 meters, a field of view (FOV) of 40° (-15° to 25°) vertically and 360° horizontally since it constantly spins. For the number of points, the VLP-32C can produce around 600,000 points in a single second, which is a very large number of points if they are all saved when scanning an area. Fortunately, the scanning software utilizes different algorithms that take care of reducing the number of saved points to decrease the file sizes. With a price of tens of thousands of euros, these high-performance LiDAR scanners are designed for the development of autonomous vehicles.

5.1 Point Clouds

When using a LiDAR or any other type of scanner that can measure distances, for example an Xbox Kinect, the area that has been scanned will become a *point cloud*. The point cloud is a database of points represented in three dimensions with, depending on the file format used, different supporting data such as a timestamp and color. There are many different formats a point cloud can be saved in but the most common, according to Autodesk [24], are Laser Areal Survey (LAS, .las), which is an industry

standard, and ASCII (.xyz). Another format is the Point Cloud Library (.pcl) which has become more popular since it is open source and free to use for commercial use [25]. The different file types store slightly different information, and different modeling software supports different formats. Once a point cloud has been loaded into an editing software, it can be manipulated, and 3D models of the point clouds can be made.

Point clouds are used in many different fields and applications, from modeling small objects to surveying large areas, and there are many different devices, scanners, that can be used to create a point cloud. If the scanner being used has the feature of saving point clouds with RGB color data, which can be achieved with an RGB-D camera for example, then an accurate colored 3D model can be created. There are other implementations of scanners that combine data from a LiDAR and a camera, which combine the accuracy of scanning with a LiDAR and the possibility to attach colors to the point cloud points when the camera takes pictures of the scanned areas. Another feature of point clouds is the possibility of measuring the distance between any two points, since the position of every point in the point cloud is known. This eliminates the need to physically having to go, for example when measuring a room, around the room with a tape measure and writing down every needed measurement in the room. With point clouds, the room only needs to be scanned once and the measuring can be done with a computer in minutes.

When using point clouds in, for example, autonomous vehicles, high-performance computers have the capability to use the point clouds to detect other cars, humans and other possible obstacles that would need to be avoided if they came in the way. However, autonomous vehicles are still under development and have quite some way to go until being approved for public use. [26]

When creating point clouds, the scanner that collects the data will, depending on the scanner, either be able to create accurate models by itself or a separate algorithm for combining the data to a useful model must be used. These algorithms are called Simultaneous Localization and Mapping (SLAM) algorithms, which will be explained in more detail in the following subsection. For example, when using the Velodyne VLP-16 LiDAR, it can only send the raw point cloud points and therefore a separate SLAM algorithm must be used if a map is needed.

5.2 Simultaneous Localization and Mapping (SLAM)

A major problem when it comes to autonomous vehicles is being able to localize where the vehicle is at any point in time, not only the global position, as mentioned before, but also the relative position to surrounding objects. This is where SLAM comes in, building a spatial map of the area around the vehicle while simultaneously having the localized position of the vehicle in the map. There are several implementations of SLAM that utilize a LiDAR, two examples that could be used for our project are either “Lidar Odometry and Mapping” (LOAM) [27] or Google Cartographer [28]. The difference between the two is that Google Cartographer is intended more for scanning indoor rooms while LOAM is more of a general implementation for scanning, but both are capable of scanning indoors and outdoors. Google Cartographer is also currently being maintained by both the Google employees and by the community that uses Cartographer. For other scanners, such as RGB-D cameras, there are other algorithms that have been developed specifically for them.

Creating maps and determining localization while standing still, or point clouds of structures that are static, is already well developed and there are robust solutions and methods for this. However, when SLAM is needed in environments that are in constant movement while mapping, and very large areas are being mapped, the complexity in the SLAM algorithms increases drastically. [29]

The SLAM problem can be explained quite easily. We have, for example, a robot in an unknown location. The robot needs to both build a map of the environment it is in, while at the same time knowing its own position on the map being built. Since the motion of the robot is uncertain, the difficulty and uncertainty increase over time while the robot moves. Having more accurate sensors to determine the environment around the robot helps significantly with the accuracy of the produced map. Within the SLAM problem there are three different paradigms of algorithms that are used when developing SLAM algorithms. They are the extended Kalman filter, a sparse graph of constraints, and a particle filter. Depending on the requirements needed for the SLAM, a different algorithm is chosen. The oldest SLAM algorithm is the Extended Kalman filter which is not as widely used as the other SLAM algorithms today, mostly due to the scaling limitations of the EKF. One problem when using a LiDAR scanner for building maps is how each scan will match to the next and how the last and first scan

will be in the same place if we start and stop in the same place, which is known as the loop-closing problem. The loop-closure problem does not only apply to LiDAR scanning, but any situation where data from a previous situation is revisited and needs to be matched for example, navigation can face loop-closure problems. Some algorithms look for common shapes in each scan and track them while building the map, for example corners and edges can be tracked to know how the LiDAR has moved during the scanning process. Another way, which is what I am trying to achieve, is to use GPS positional data to increase the accuracy of the loop-closing. Adding an IMU will also help with knowing the orientation of the LiDAR and thus being able to better track its movements. [29]

6. Implementation

As mentioned in the introduction, the solution will be comprised of three sub-systems running simultaneously to keep track of orientation, GNSS data and the filter that will combine the GNSS with the IMU data to output accurate location data. The final system will output orientation data in quaternions and accurate location data in latitude, longitude and altitude. The orientation is needed for the LiDAR and the location data is needed for knowing where the vehicle or LiDAR has been and for building the georeferenced point cloud. The system will be a part of a larger system where the INS will be started by another program.

6.1 INS construction

To construct an INS, the main components needed are an IMU, a GNSS receiver and a computer to use the received sensor data and calculate the current location. For the solution I have decided to use at least two IMUs, based on previous research [6, 7], and a GNSS receiver that is able to obtain data from GPS and GLONASS, as well as Galileo satellites. For the computer, a Raspberry Pi 3 or Odroid C2 will be used since they have more than enough computational power for the solution and are easy to develop with. When it comes to software, the main platform that will be used is the ROS, more specifically ROS Kinetic. The reason for using ROS is that the software that creates the point clouds has been developed in ROS and needs the IMU data to be sent within the ROS environment. Moreover, the LiDAR system project started its development when ROS Kinetic was the most relevant version of all used ROS packages, thus Kinetic was also used for the INS project. Another reason for using ROS is that it is easy to learn, and many examples can be found online. ROS is also a good platform when it comes to debugging and troubleshooting, since there are ready data visualizer programs that can help with finding problems in the output of our program. When developing in the ROS environment, there are many programming languages that can be used, since ROS is already implemented in C++, Python and Lisp [32]. For the custom solution being built, I have developed the different sub-systems in C++. The main algorithms used are the Madgwick AHRS filter or

complementary filter, depending on the performance, MS AUKF and a basic GNSS message parser.

6.2 Design of the INSS test platforms

I have designed the INS hardware setup in the following way. The IMUs are connected to the Raspberry Pi 3 or Odroid C2 using an I^2C connection and the GNSS module is connected using a serial connection, as shown below in Figure 5. In the initial test phase I noticed that there were some problems powering the GNSS module directly from the Raspberry Pi USB pins, which resulted in the Raspberry Pi becoming unresponsive and crashing. This was fixed by switching over to the Odroid C2 and reinstalling the operating system, Ubuntu Mate. For the IMUs, I started with searching for low-cost IMUs that were available to purchase and made a list of the different performance metrics that were available in the datasheets. After comparing all the IMUs, I chose the Pololu MinIMU-9 v5 [38] based on price, performance and availability, there were many different IMUs to choose from. The Pololu MinIMU-9 v5 has an ST LSM6DS33 accelerometer and gyroscope module and an ST LIS3MDL magnetic field sensor module. The GNSS receiver I chose, a Ublox Neo-M8N, was selected based on availability and performance. The receiver was connected to the Odroid C2 through the serial Tx/Rx pins on the Odroid. When performing the different tests, the Odroid was powered by a power bank. The entire hardware architecture for the custom system can be seen in Figure 5, including the different communication protocols the components use.

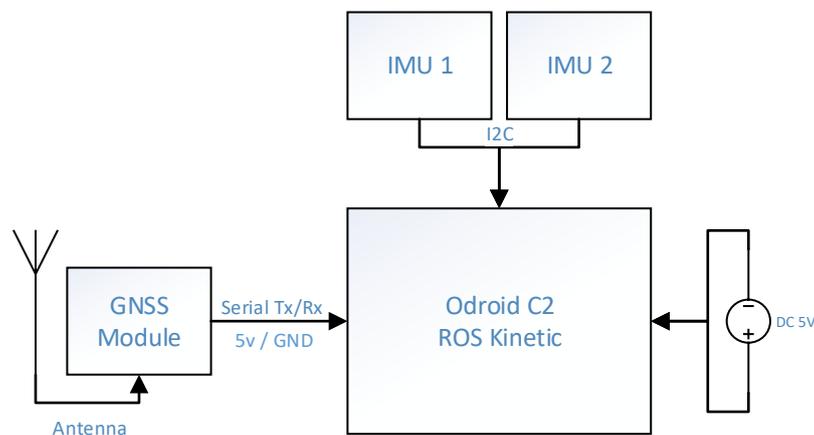


Figure 5: The hardware architecture of the INS/GNSS system.

6.3 Design of software

To create the proposed solutions, the different sub-systems including IMU nodes, combine node, GNSS node and INS node, were programmed and tested one by one, starting with the IMU code. The IMUs I decided to start out with were the two Pololu MinIMU-9v5 [38] which had ready test software available from GitHub. Since the code was free to modify and use, I decided to modify the code to work with ROS so that the raw IMU data is published in ROS. Additional code was also needed to offset the IMU data to the center of mass of the system [21]. Fortunately, ROS has a ready package that can transform movement data according to a user-defined offset and I decided that this was the most reliable implementation to use. Next, I wrote the code to combine the two IMUs, this time starting from scratch. The code simply subscribes to the IMU data topics that are published by the IMUs and takes an average of the two IMUs and publishes the new combined data to ROS. In addition, it makes sure that the ROS messages being received were synchronized to have the most accurate average of the two IMUs.

In the future, if necessary, one can consider implementing a better way of combining the data from two or more IMUs. Different ways of combining data from IMUs has been tested and shown greater accuracy [6]

Once the IMU data worked and had been tested, the custom Madgwick orientation filter implementation was coded as a ROS package. ROS also has a number of different AHRS filters already implemented, for example the complementary filter has its own ready implemented package [34]. To compare the Madgwick solution to the complementary filter which is already available as a ROS package, tests of how well they tracked orientation were done by me and I chose the one that performed best. The tests that were conducted showed that the complementary filter was the most accurate with the IMU data and thus chose to use it over our own Madgwick implementation.

When all the IMU parts of the system were done, the GNSS module and GNSS message parser was next. A GNSS message parser was implemented using one of the available C++ libraries. The library chosen [45] had to be slightly modified since it was designed to receive GPS messages, while the Ublox receiver will be receiving combined messages with a slightly different format. The parser receives National

Marine Electronics Association (NMEA) messages from the GNSS receiver and parses the messages to a ROS navigation message, NavSatFix message, that includes latitude, longitude, altitude and velocity. If other data, such as the number of satellites being tracked, are needed, these can also be obtained from the parser, as long as the correct NMEA messages are received. Two separate GNSS message parsers had to be implemented, one for the custom INS solution and one for the Applanix APX-15 INS, since the APX-15 used a USB connection and the custom INS was wired to the serial Tx/Rx ports. Moreover, the ROS topic messages had to have different names if I ever wanted or needed to have both systems running at the same time. It is not possible to run two ROS nodes with the same name or having two ROS topics with the same name since they would be conflicting with each other.

The most time-consuming part of the whole system was coding the MS AUKF filter [20]. There were some ready UKF implementations, but none had the adaptive part that improves on the regular UKF [35]. This had to be added to an existing UKF implementation, since it was easier than implementing the entire filter from scratch. The code that I needed to add was the slave part of the MS AUKF, which is a UKF filter that takes the noise covariance matrix and adds white noise for every time update and the innovation from the master filter for every measurement update. The innovation is the calculated error between the predict and update measurements, which is simply calculated by subtracting the estimated value from the updated real, correct, measurement value. The equations needed to implement the slave filter can be found in [20]. Because computational power is not something that has to be taken into consideration in our solution since no limitations on the size or power consumption of the INS has currently been set, the slave filter will be implemented using the UKF. Once the MS AUKF filter was coded, the filter could be tuned to achieve better performance. This had to be done by testing and changing the different constants. The entire software architecture of nodes in the custom system can be seen below in Figure 6.

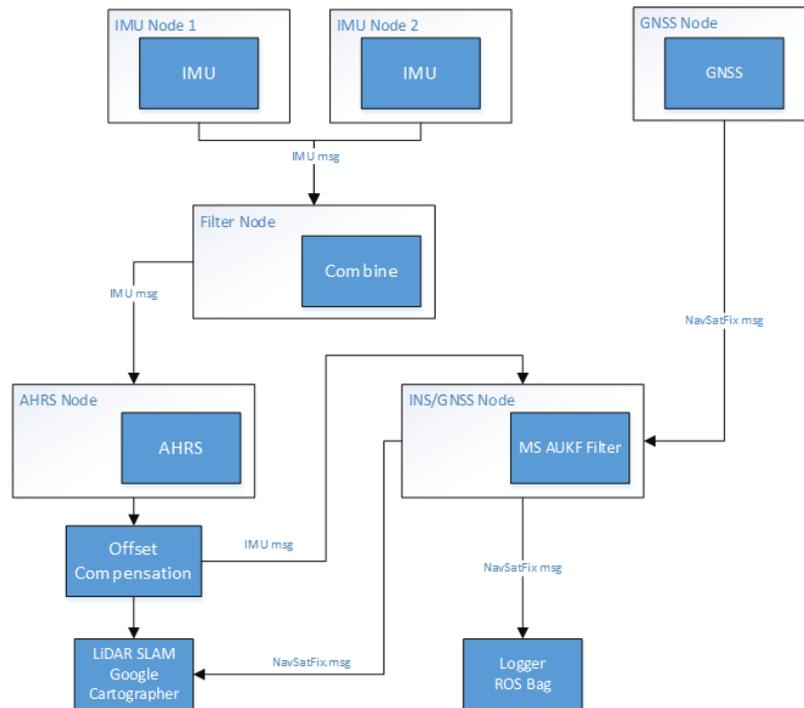


Figure 6: The software architecture of the INS/GNSS.

When configuring the custom INS system, the IMUs and combine filter were configured at 100Hz and the AHRS filter was also configured to 100Hz. The GNSS receiver was not configured, the factory settings were used, but the parser, however, was configured to receive messages at a rate of 10Hz.

Before starting the experiments, tests were made to compare the different AHRS filters, as well as how they perform when having the magnetometer enabled and disabled. What was discovered was that, in the custom INS system, the magnetometer affected the filter in a negative way, either there was a delay after rotating the system, or the heading of the system moved but returned to the initial position after a few seconds even though the system had not been moved back. When testing the AHRS filter without the magnetometer, the orientation tracking was noticeably better at keeping track of the system orientation movements, thus it was decided to keep the magnetometer disabled. This may change in the future if it is decided to change out the IMUs and retest the filters with the new IMUs.

6.4 Commercial system: Applanix APX-15

The Applanix APX-15 UAV is a precision GNSS-inertial solution designed for small UAVs and intended to be used for mapping and image georeferencing. The APX-15 supports all GNSSs that are available, accessing a total of 336 GNSS channels, capable of raw GNSS data rate of 5Hz, IMU data rate of 200Hz, with the possibility of a maximum 100Hz data rate for the accurate sensor-fused positioning data which is fused using a Kalman filter. It has an accuracy of 0.5-2.0 meters when using DGPS and is capable of utilizing RTK for increased accuracy. The receiver also has post-processing capabilities that increase the accuracy to 0.02-0.05 meters. With a price of around \$18,000 and the whole system on a single board, the APX-15 UAV is a capable system that is a good reference to compare to the custom INS. The custom system will be set to output the fused location data at a rate of around 30Hz and I will have to configure the APX-15 to have the same rate to have a fair comparison. [33]

For the APX-15 I needed to use another Odroid C2 and connect a serial to USB converter in between since they were further apart from each other and having a single cable was easier than having four. The hardware architecture can be seen in Figure 7 below.

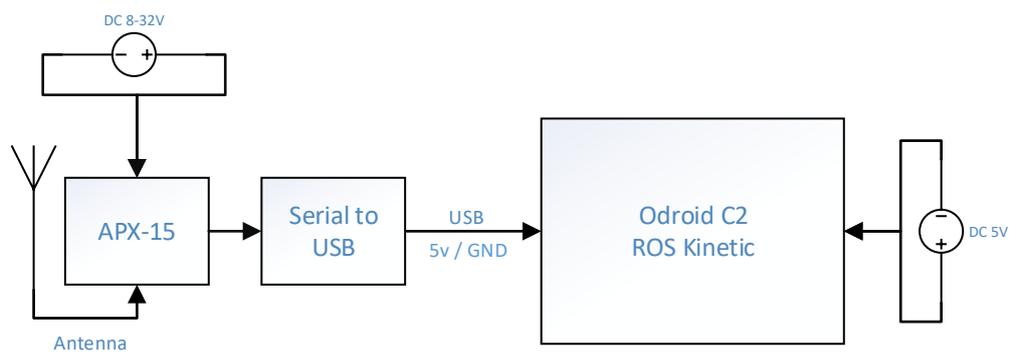


Figure 7: The hardware architecture of the Applanix APX-15 with logger.

7. Evaluation

In the evaluation phase, two different experiments were designed and performed. The first experiment was a vehicle experiment where the system was mounted on either a bicycle or in a car to compare to a commercial solution. The second experiment was the LiDAR device / Point Cloud experiment where the system was mounted on or close to the LiDAR device while the LiDAR was scanning the surrounding area to figure out if it improves the loop-closing. I shall begin with explaining the reason for the experiments, the practical experiment setup, the hypothesis of the experiment, the experimenting process and finally validating the results. The measured results from the experiments are shown graphically for easier analysis. Once the data has been analyzed and a conclusion drawn from the analyzed data the performance of the system will be known.

7.1 Research question 1: INS vehicle experiment

The reason for this experiment is to both test the system functionality and to compare the system performance to the Applanix APX-15. Since I were to compare the custom system with the Applanix APX-15, I first defined a path to take and for each system a tester at Brighthouse Intelligence drove around the path while logging data. The initial plan was to have both systems running at the same time next to each other, but due to the lack of two identical antennae, the systems were tested separately.

7.1.1 *Experiment setup*

For the experiment setup on a vehicle, the two IMUs and the GNSS receiver were mounted on a 3D-printed test board, and the experiment board was rigidly mounted to the vehicle. As mentioned in the previous chapter, the system uses an Odroid C2 which was mounted next to the experiment board and used to compute the accurate location. All software necessary for the INS to work was running in the ROS Kinetic environment. ROS has a built-in function that can record selected messages to a ROS bag, which is a log file containing all messages that were selected to be recorded. The

data that was being sent between the different nodes and outputs from the nodes during the experiments were saved to a ROS bag, which can be played back at any time.

The Applanix APX-15 is a standalone system that can log data to an external drive or internal memory, or it can output NMEA messages over serial or over a network. In the test the serial data connection was used. For simplicity, the APX-15 was connected to the same Odroid as the custom INS system, which would make simultaneous testing possible in the future. Before testing the APX-15 it had to be calibrated and different offset measurements had to be configured through its web interface.

Two different antennas were used in the experiment, one that came with the APX-15 made by Tallysman and one that was purchased from a Chinese webstore and which manufacturer is unknown. The following antenna specifications were listed on the store page: bandwidth of 1570.42-1580.42Mhz (L1), Voltage Standard Wave Range (VSWR) of < 2.0 , LNA gain ≥ 27 dB, noise figure of ≤ 1.5 dB and interference suppression of 20dB [44]. The Tallysman has the following antenna specifications: bandwidth of 1213 – 1261Mhz (L2) and 1557 – 1606Mhz (L1), VSWR of < 1.5 , LNA gain of 35dB, noise figure of 1.5dB and Out-of-band rejection of 30 – 40dB [43]. The major differences between the antennas are the bandwidths and gain. The Tallysman antenna is able to receive signals on both L1 and L2 band while the Chinese antenna is only able to receive signals on the L1 band. The bandwidths also reveal the fact that the Chinese antenna is only able to receive signals from GPS and Galileo while the Tallysman is able to receive signals from GPS, GLONASS and Galileo. The Tallysman also has a much higher LNA gain than the Chinese antenna.

7.1.2 Experiment hypothesis

With this experiment the following hypothesis will be verified: A low-cost INS can be built to provide reasonable quality and positional accuracy that is comparable with a commercial system, in situations currently required by the LiDAR system where the INS will be installed.

7.1.3 Experiment Process

The experiment started with planning a route on a map which was to be followed as closely as possible. Once the route to be taken was defined, the vehicle was moved to the starting position and the systems were turned on. Before leaving, the systems were checked, they were both functioning and that the GNSS receiver had a lock. When everything had been initialized the logging of data was started, a ROS bag began to record, and the vehicle started to travel along the planned route. If there were any deviations during the trip, notes would be taken so that the detour could be verified in the analysis. Once the vehicle had reached its destination the logging process was stopped, and the data saved to an external drive was transferred to another pc for analysis. When analyzing the data, the logged data from the two devices were overlaid on a map to see how accurate the data is compared to the planned route that was taken, as well as comparing the two system outputs.

The used experimentation vehicle was a bike with the INS systems in a box strapped to the rear carrier of the bike, making sure the INS systems were well mounted inside the box and the screen visible through the box. Figure 8 shows how the box was mounted to the bike.



Figure 8: Test bike with test setup on the rear carrier.

The experiment path was pre-planned, to include sharp turns and long straights, using Google Maps, by drawing the path on a map. The path was followed by remembering the path by heart and always staying on the right side of the road. The map of the full path can be found in Figure 9 below. The length of the path was about 2 km which is not a long distance to travel by vehicle, however, it is perfect for the current use case of the LiDAR system which the INS will be mounted to.

The test data maps were produced using ROS Mapviz, which uses maps produced by Google Maps, by subscribing to the fused NavSatFix message topic and drawing points or lines on the map.



Figure 9: The full planned route taken, about 2 km in total length.

Since the INS will be used in environments where the GNSS antenna will not have perfect visibility of the sky, the path includes a more challenging part where sky visibility is limited, and rapid turns are taken. The challenging part can be seen below in Figure 10, where trees are blocking the view of the sky.



Figure 10: The most challenging part of the route where trees block the sky and several turns are taken.

7.1.4 Experiment results

Figure 11 below displays the map resulting from when testing the Applanix APX-15 and Figure 12 displays the close-up of the challenging part of the path. The resulting tracked path is very accurate, only minimal drift can be seen when the APX-15 traveled through the challenging part on the way back to the starting point. Another small drift has occurred when the test started, and the bike drove down a ramp from the parking lot. Both INS devices used the antenna that came with the APX-15.



Figure 11: The APX-15 full path.



Figure 13: The custom INS full path.



Figure 14: Close up of the challenging part with the custom INS.

The two initial tests were performed to verify the functionality of both systems as well as to be able to compare the results with the two next tests in case having two INS devices installed next to each other caused interference.

For the next test, both systems were in the same box and the antennas used were the ones that came with the respective systems. Figures 15 and 16 below show the resulting tracked paths from both systems.



Figure 15: Both systems in the same box, APX-15 using its antenna and the custom INS using its antenna. The pink line is the APX-15 and the green line is custom INS.

In Figure 15 the custom INS has drifted much more than in the initial test using the APX-15 antenna. This is understandable considering the difference of the two antennas. The challenging part, Figure 16, is also much worse than the initial test, drifting more than 10 meters on the way back. This is most likely due to the different antenna used in the systems.

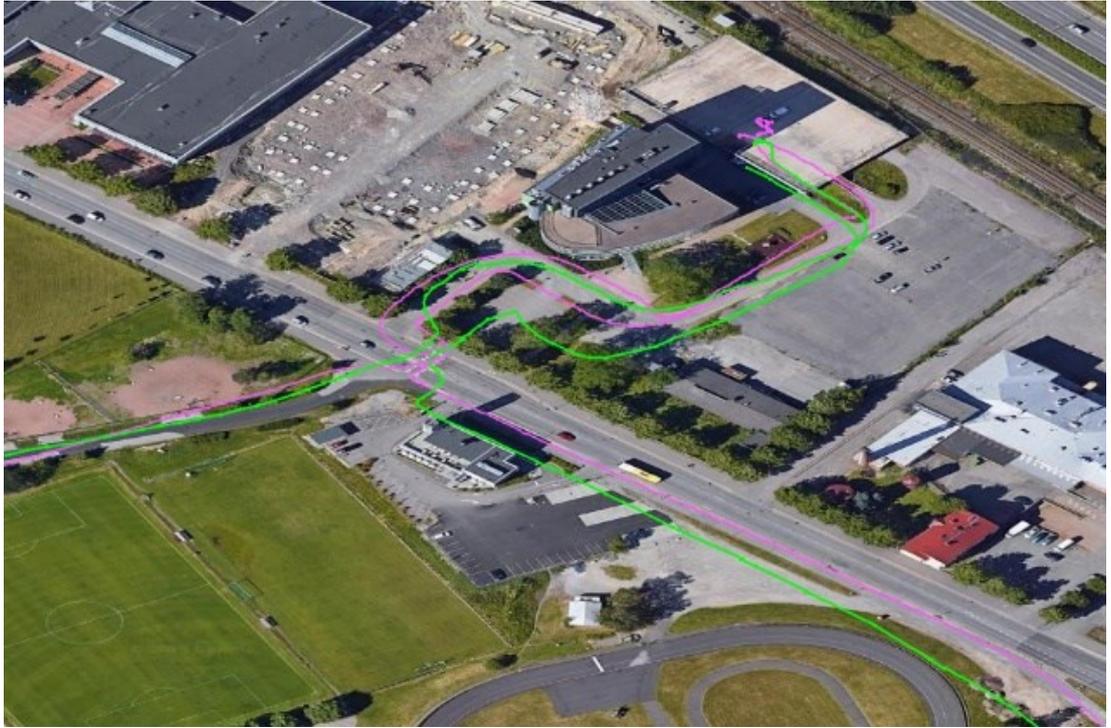


Figure 16: Close up of the challenging part with both systems.

In the following test, the antennas were switched. Figures 17-18 are the resulting maps.



Figure 17: Both systems, APX-15 using the Chinese antenna and the custom INS using the APX-15 antenna. The pink line is the APX-15 and the green line is the custom INS.



Figure 18: Close-up of the challenging part, both systems' paths drawn.

Figure 17 shows how the APX-15, pink line, has drifted much more than the custom INS, green line, both in the more challenging place as well as by the large oval ring. Figure 18 shows how much more the APX-15 has drifted in the more challenging place, though not as much as the custom INS when tested with the same antenna, in Figure 16. The custom INS is, however, in no way perfect, drift has occurred around the start areas as well as after the challenging place.

Comparing the two INSs when using the same antenna, Figures 16 and 18, we can see that the APX-15 has performed with minimal drift and error while the custom INS has had slightly more drift when the trees blocked the view of the sky. When analyzing the figures, the APX-15 had a maximum error of 1-2 meters while the custom INS at the least accurate place resulted in a maximum error of 5-6 meters.

Analyzing the data from the test where we both systems have their own antennae, we see that the APX-15 has again a nearly perfect localization while the custom INS, using its own antenna, has drifted quite a lot in the right corner of Figure 16 which can also be seen in Figure 15. The error is corrected once the route became straight, and the accuracy is even kept quite good when looking at Figures . After switching around the antennas, we notice how the antennas affect the accuracy of the two INSs. The APX-

15 performs noticeably worse with a Chinese antenna and the custom system has performed noticeably better with the antenna by Tallysman. This most likely due to the Tallysman having a larger bandwidth and L2 band. Comparing Figure 14 with Figure 18, the custom INS has performed much better in the second test than the first test. This could be caused by satellite availability or the test system had problems. Comparing Figure 19 and 20 side by side, the accuracy difference between the different systems using different antennas is clear, the APX-15 has drifted less overall.



Figure 20 Pink line is APX-15 and custom INS is the green line, using own antenna.



Figure 19 Pink line is APX-15 and custom INS is the green line, antennas switched.

Ideally, the systems would use the exact same antennas at the exact same time but, unfortunately, only one of each antennae were available to be used during the device testing. In the tests we notice how much the choice of antenna affects the GNSS/INS performance.

If we compare Figure 16 from the APX-15 test and Figure 18 from the custom INS test, the road leading to the building has a decreased accuracy. This decreased accuracy is most likely due to the road having on both sides trees which block the road from direct view of the sky, thus resulting in a weaker satellite signal.

7.1.5 Experiment conclusions and observations

The experiment showed that the custom INS was able to track the movement of the bike to an accuracy nearly as good as the APX-15. However, this was achieved when using the antenna that came with the APX-15 and not the one that came with the GNSS receiver used in the custom INS.

For testing both INSs at the same time, it was not possible to have the logging on the same Odroid C2. When tests were made using only a single Odroid, the node which published the fused location data as a ROS NavSatFix message kept crashing and I decided to use two Odroids at the same time to log the INS data. The ROS bags could then be played back simultaneously on a separate computer where the map images were created. Moreover, to test the systems in a commercial system versus the custom low-cost system, I decided that the APX-15 would use its own antenna and the custom system would use a basic antenna that the low-cost GNSS receivers come with.

During the experiment, multiple tests were run where the custom INS was tested, and the route was taken going both clockwise and counter-clockwise. The final tests which are in the results were done once the custom INS was proven to work reliably and without problems.

7.2 Research question 2: Loop closing improvement

For this experiment, the objective was to see if the loop closing problem was solved or improved when using an INS in combination with the LiDAR. The end product of the experiment was a point cloud that could be used to obtain different measurements of the house which was scanned, as well as storing accurate location data. The place being scanned with the LiDAR was according to the use case of the LiDAR scanner, which is currently larger structures such as houses. This was the main test for the system performance since the entire system was tested.

7.2.1 Experiment setup

Similar to the INS/GNSS vehicle experiment setup, the same hardware and software was used. The scanning device was unfortunately under development at the time when this experiment was conducted and could not be shown in its entirety, only a description of the device can be given. The scanner consisted of a Velodyne VLP-16 LiDAR scanner, the same IMUs as in the vehicle experiment, and a Nvidia Jetson TX2 embedded computing device. The hardware architecture of the experiment system can be seen in Figure 21 below.

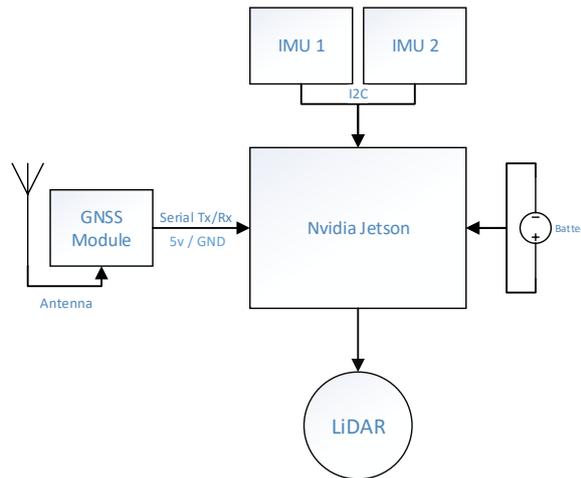


Figure 21 Hardware architecture of the LiDAR scanner unit being developed.

Since the custom INS uses the ROS environment and so does the LiDAR system, we could easily record a ROS bag where all data that was published within ROS was saved. When comparing using location data with not using location data, the LiDAR and INS topics the ROS bag will be played back while using a SLAM algorithm to build the point cloud. This would allow the experiment to only require a single scan to be performed. When playing the recorded ROS bag, the SLAM algorithm was either to be configured to use location data or not.

7.2.2 Experiment hypothesis

Since the LiDAR has an accuracy of a couple of centimeters at 100 meters, attaching an INS with an accuracy of up to 1 meter to improve the loop closure of point clouds might not be possible. However, the INS could help with combining scans to each other if a further distance had been traveled and the system then returned to a point already visited. This does not depend on the LiDAR scanner but the loop closure algorithm being used within the SLAM algorithm. In this situation the INS could improve the final point cloud. In the experiment, the area being scanned was of a house, which had many features that could be used to track odometry and be used for loop closing. However, there could be a possibility that the SLAM algorithm loses track of the features being tracked, in this situation the INS could help the SLAM algorithm.

7.2.3 *Experiment Process*

The experiment consisted of the following steps. Starting outside in a predefined position we turned on the custom system as well as the LiDAR system. Once both systems were up and running, the ROS bag started recording the topics being sent within ROS and we started to walk along the predefined path. Since the main problem that the IMUs are supposed to solve are quick movements of the LiDAR while scanning, as well as the improved loop-closing, we would at random points quickly move the LiDAR scanner around to best test the custom system. When the scanning was complete, the ROS bag was saved and transferred to a different computer where I could compare the point clouds from having the custom INS solution on or off. Because we had saved the INS data to a ROS bag, I could use it with Google Cartographer to first create a point cloud having the SLAM algorithm not listening to the INS messages, and then enable the SLAM algorithm to use the INS messages.

The LiDAR driver and the custom INS software was run simultaneously without having a SLAM algorithm running, due to tests that showed that there was possibly not enough computational power left for both processes.

7.2.4 *Experiment results*

In the experiment, I obtained two different point clouds, one with the INS data enabled and another one without INS data. These point clouds were imported into CloudCompare [39], which is an open source software for viewing and processing point clouds. Unfortunately, the GNSS was unable to receive a lock when we started from inside the building walking out, the receiver needs to have line of sight to the sky when it is powered on to be able to lock in an acceptable time period. Figures 22-23 were obtained after building the point cloud with the SLAM algorithm not using location data. Figures 24-25 were obtained after building the point cloud with location data enabled.

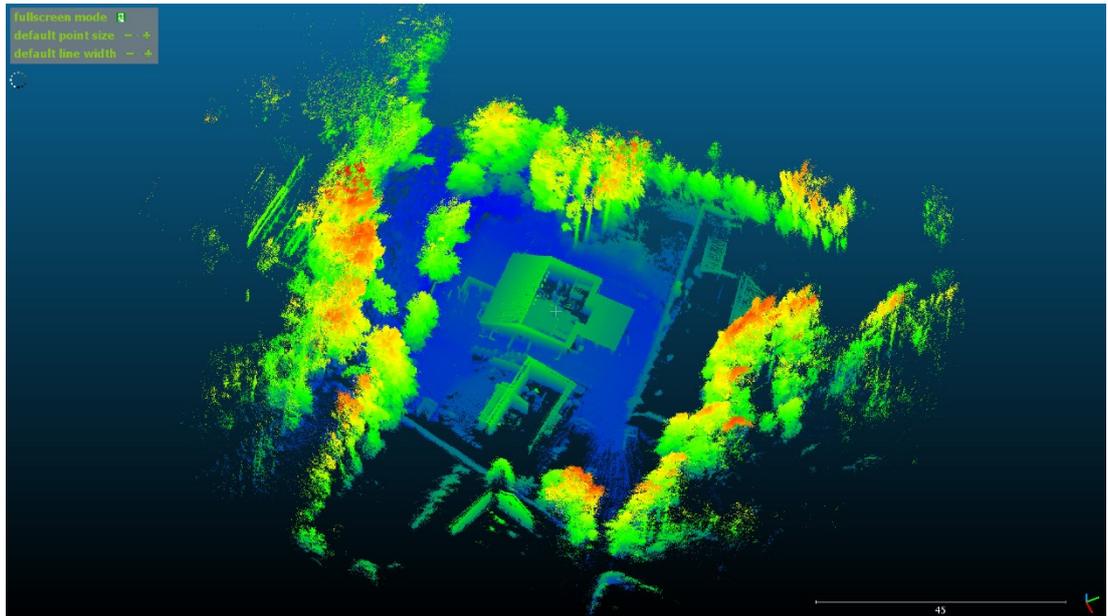


Figure 22 Point cloud created without enabling INS messages in the SLAM algorithm used to build the point cloud.

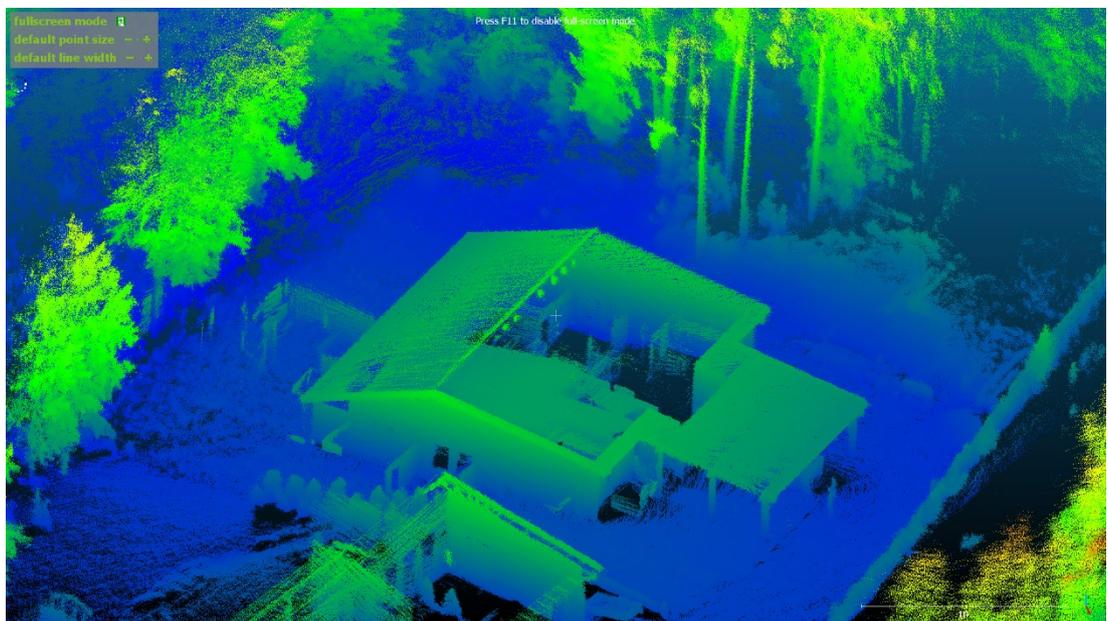


Figure 23 Zoomed in picture of the house that was scanned during the experimentation, INS messages disabled.

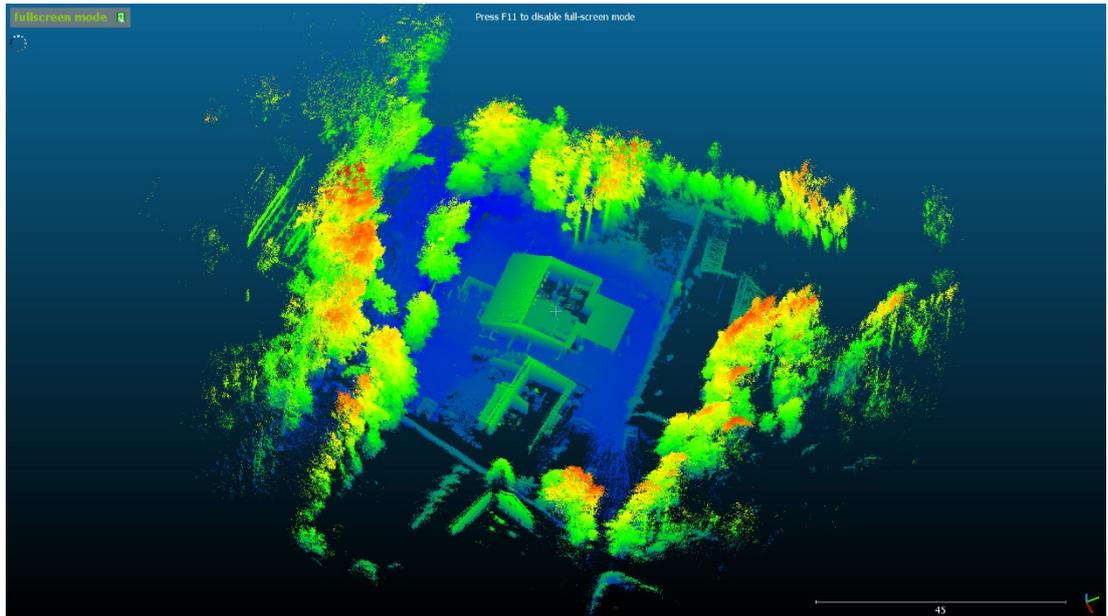


Figure 24 Point cloud built using INS messages enabled in the SLAM algorithm.



Figure 25 Zoomed in image of the house that was scanned in the experiment, INS messages enabled.

From the Figures 22-25 above, no noticeable difference can be seen, the INS has had little to no effect on improving the quality of the point cloud, thus the loop-closing improvement hypothesis is confirmed.

7.2.5 *Experiment conclusions and observations*

Having tested the LiDAR scanner device in cases where the device will most likely be used in, there were no noticeable loop closure advantages. The algorithms that were used for solving the loop closure problem in the experiment were good enough. There are situations where the INS will have more effect on the loop closure which were not tested due to there not being a current use case for the company. The loop closure will possibly improve when scanning less feature rich environments such as open fields or when driving on a highway. In these situations, there are less ways for the LiDAR to track odometry and it will thus benefit from having an INS and location data when building the point cloud. Tests of a scan had shown positive results when using an INS while scanning a more open area without features that can be used for odometry tracking and loop closure [36].

8. Summary of work

The main work that has been done has been implementing IMUs to help with reducing the error in a LiDAR scanner caused when moving around with the scanner while scanning. Moreover, an INS was implemented to achieve an accurate georeferenced point cloud for knowing where a scan has taken place, by plotting location data on a map, and for future use cases. The system code was written in C/C++ and was implemented in the Robot Operating System, so that it could be used for different projects and to increase the modularity of the system. The total price of the custom INS was around 100€.

Two different experiments were conducted to test the system. The first experiment was to test the functionality of the system and to see that the custom system functioned according to the requirements. The second experiment was conducted to see if there were any benefits from using an INS when building point clouds, from the current use case perspective. From the experiments I was able to conclude that the system worked according to the requirements and that there were no noticeable improvements in the quality of the point cloud when using INS data. When comparing the custom INS to a commercial one, I noticed that the location accuracy depends highly on the quality of

the antenna being used. Thus, if accurate location data is needed, buying a better antenna is a good start and worth the price.

Lessons that have been learned from implementing, testing and evaluating have been many. When starting to learn the new ROS framework, it took a long time to understand how the different parts were connected and how to implement new nodes. With ROS, it is best to take one's time to learn the basics and look how ready implemented code is structured. This ensures a good base knowledge of the system and makes it much easier to implement new software. A more hardware-based lesson learned is that components might not always work as expected and implementing simple tests can reduce debugging time significantly. As an example, the GNSS receiver did not output any location data during initial component testing, which I spent a lot of time debugging before realizing that the receiver will only obtain a location lock when powered on outside. This made early tests inconvenient because I had to bring a keyboard and mouse outside to be able to start the tests. Later I was able to use a touchscreen connected to the Odroid and scripts that started the testing software.

My work will help Brighthouse Intelligence to increase their point cloud accuracy and create more use cases where a scanner can be implemented. The most important development made was the IMUs being able to eliminate errors, caused by movement of the LiDAR, from the point cloud. The software for the IMUs can be used for different projects, for example an IMU can be implemented with a camera scanner to increase its accuracy and reduce movement errors that may occur, or in vehicle navigation. Moreover, the INS can also be used for any other project that might need accurate location data and in cases where only non-fused GNSS is needed, there is ready implemented software for that too. This is the advantage of having a modular system. There are many different ways that the custom system can be improved and expanded, examples of improvements will be discussed in the next section.

9. Future work

Point clouds and LiDAR technology in general are both relatively new technologies which are constantly being developed for many different industries. Therefore, there are many use cases that have yet to be discovered. There are many ways to improve the system I created, both in terms of hardware, as well as in discovering new use cases. The INS system does not necessarily have to be implemented directly into ROS, it could be developed into its own standalone system that only sends fused GNSS location data over serial, the same way as the Applanix APX-15 does. More accurate IMUs could also be bought and test how much more accurate they are, both at improving the walk movement error in the LiDAR, as well as increasing the INS accuracy.

In terms of software, there are many different INS and IMU algorithms that have not been tested. These algorithms may include more sensors, such as a barometer or wheel encoders, to increase the accuracy of the odometry. When combining the two IMUs, instead of taking the average of the two I could implement a Kalman filter that filters the data in a more accurate way. Moreover, the LiDAR data from the scanner could be used to calculate odometry and fuse it with the INS to increase the accuracy of the positional data even more.

Swedish summary

Inledning

Att navigera och veta var man har färdats har länge varit ett problem, både på land och till havs. På land kan man med hjälp av kartor och landmärken lättare navigera, men till havs har man ett problem i och med att det saknas landmärken. Olika instrument har utvecklats för att hjälpa till med att navigera, till exempel sextanten som använder solen och stjärnorna för att beräkna positionen man befinner sig i. Med introduktionen av satelliter som kan användas till navigering har man kunnat förbättra navigeringens noggrannhet drastiskt, och möjligheten att utnyttja mera noggranna platsdata till andra ändamål än navigering ökar. Då man får platsdata från en satellitmottagare sker det oftast med en frekvens på en hertz, med dyrare mottagare upp till tio hertz. Problemet med att få nya data med en frekvens på en hertz är att ifall man färdas med en hög hastighet eller rör sig mycket sporadiskt kommer de platsdata man får att vara mera utspridda. Genom att implementera och använda en tröghetsnavigator kan platsdata förbättras och man kan få nya data med högre frekvens.

En tröghetsnavigator använder data från accelerometrar, gyroskop, magnetometrar och satellitmottagare. Genom att tillämpa datafusion på data från sensorerna kan man få noggranna platsdata med en högre frekvens. Oftast är utdata från tröghetsnavigatorn i form av latitud- och longitudkoordinater för att man lättare ska kunna använda dessa data med kartor. Ett användningsområde som blivit allt vanligare är att kombinera en tröghetsnavigator med en LiDAR (eng. *Light Detection and Ranging*) för att få en 3d-representation av omvärlden i formen av ett georefererat punktmoln. Dessutom finns det möjligheter att utnyttja platsdata då man bygger punktmolnen, vilket resulterar i ett mera noggrant punktmoln.

Syfte och ämnesmotivering

Syftet med detta diplomarbete är att undersöka möjligheten att implementera en förmånlig tröghetsnavigator som är jämförbar med ett mycket dyrare tröghetsnavigatorsystem. Tröghetsnavigatorn kommer att implementeras modulärt, så att data kan användas från alla de ingående sensorer för olika ändamål. Dessutom undersöks möjligheten att använda data från min tröghetsnavigator för att öka på

noggrannheten av ett punktmoln. Diplomarbetet är ett uppdragsarbete av Brighthouse Intelligence Oy, som är ett mindre företag, specialiserat på marinteknologi och automation. Brighthouse Intelligence hade som mål att undersöka ifall man med hjälp av relativt billiga komponenter och sensorer kunde bygga en tröghetsnavigator som kan användas med en LiDAR. Syftet är också att få en bättre förståelse för hur punktmoln byggs, hur de kan förbättras samt undersöka nya användningsområden inom vilka man kan dra nytta av punktmoln. Under arbetets gång undersöktes även andra sensorer och metoder som kan användas för att bilda punktmoln och öka tröghetsnavigatorns noggrannhet.

LiDAR-sensorer har under de senaste åren utvecklats med enorm fart på grund av utvecklingen av autonoma bilar och andra fordon. Genom denna utveckling har man dessutom upptäckt många nya användningsområden där en LiDAR kan användas, bland dem möjligheten att skapa detaljerade modeller av byggnader och andra objekt utan att använda måttband och bilder. Genom att använda en LiDAR för att skanna ett hus eller ett område sparar man mycket tid, både när man utför skannandet och då man arbetar med att få fram 3d-modellen av de redan skannade objekten. LiDAR:n eliminerar behovet att ta enskilda mått och man får en färdig modell på datorn som går att bearbeta vidare.

Noggrannheten av punktmoln beror mycket på de algoritmer som används i byggandet av punktmolnen. Algoritmerna för att bygga dessa moln kallas för *samtidig lokalisering och kartläggning* (eng. *Simultaneous Localization and Mapping SLAM*). Algoritmerna använder olika sätt för att sammanfoga enskilda punkter till ett punktmoln. Till exempel kan algoritmen känna igen kanter och hörn av objekt som går att använda som landmärken då punktmoln sammanfogas.

Problemet med att endast använda en LiDAR då man skannar ett objekt är att den inte är så bra på att spåra snabba rörelser. Detta leder till dåliga punktmoln, speciellt då man rör sig snabbt och sporadiskt medan man skannar. Dessutom behöver SLAM-algoritmerna landmärken som det möjligtvis inte finns så många av då man rör sig på öppnare områden. Algoritmerna kan utnyttja andra sensorer, såsom accelerometrar, gyroskop och magnetometrar, för att hjälpa till med spårandet av rörelserna LiDAR:n utsätts för medan man skannar. Sensorernas noggrannhet spelar en stor roll i hur mycket noggrannare punktmolnen blir.

Presentation av metod och material

För att få ett georefererat punktmoln behövs två system, en tröghetsnavigator och ett LiDAR-system. Brighthouse Intelligence hade redan implementerat LiDAR-systemet, därmed behövdes endast en tröghetsnavigator implementeras. Tröghetsnavigatorn skulle dessutom kunna ge två olika utdata, LiDAR-systemets ställning i jämförelse med jorden samt noggranna platsdata. Ett annat krav var att systemet skulle fungera med robotoperativsystemet ROS som är ett gränssnitt som fungerar genom att exekvera olika noder i operativsystemet Linux. Fast det heter robotoperativsystem så betyder detta inte att det bara kan användas till robotar, utan ROS kan användas till vilket system som helst som behöver skicka mycket data mellan olika noder.

Jag använder ROS för att lättare kunna skicka data från tröghetsmätaren till LiDAR systemet. ROS fungerar genom att man programmerar olika noder som antingen tar emot eller skickar data, eller båda. Noderna bearbetar sedan datan på det sätt man vill. I mitt system finns det totalt sex stycken noder. Båda tröghetsmätarna har varsin nod som skickar sensordata till en annan nod som kombinerar dem och skickar resultatet vidare till AHRS-filtret. Från AHRS-filtternoden skickas datan vidare till nästa filternod som fungerar som tröghetsnavigator. Satellitmottagaren har också en egen nod som skickar platsdata till tröghetsnavigatornoden. Tröghetsnavigatornoden mottar och bearbetar alla data, beräknar datafusionen och skickar sedan noggranna platsdata till LiDAR-systemet.

Jag började med att ta reda på vilka sensorer som kunde ge ställningen för enheten och vilka sensorkomponenter det fanns att köpa från närmaste elektronikbutik. Till slut beslöt jag mig för att köpa två Pololu MinIMU-9 v5 som är tröghetsmätare (eng. *Inertial Measurement Unit, IMU*) som har en 3d-accelerometer, 3d-gyroskop och 3d-magnetometer. Dessutom köpte jag en satellitmottagare som kan motta data från alla satellitsystem som finns, en Ublox NEO-M8N.

När jag hade alla komponenterna kunde jag börja programmera noderna till tröghetsmätaren för att få ställningen som enheten är i och kunna spåra hur den rör sig. Detta kan man göra genom att implementera datafusion på de data som kommer från accelerometern, gyroskop och, ifall man vill, även magnetometern. Ett sådant system kallas ställnings- och riktningsreferenssystem (eng. *Attitude and Heading*

Reference System). En algoritm för att få ställningen som är enkel att implementera är ett komplementärt filter (eng. *complementary filter*). Filtret fungerar så att man har ett lågpasfilter och ett högpasfilter där data från accelerometern passerar högpasfiltret och data från gyroskopet passerar lågpasfiltret. När man använder tröghetsmätare finns det även mera invecklade och utvecklade algoritmer för datafusion, dessa algoritmer kommer att experimenteras med för att hitta den som fungerar bäst i systemet som byggs.

När datafusionen i tröghetsmätaren har prövats och en av algoritmerna har valts, kan nästa datafusionsalgoritm implementeras, mellan AHRS-filtret och satellitmottagaren. Idén med att kombinera dessa två sensorer är att man får veta hur LiDAR-systemet, eller tröghetsnavigatorn, har rört sig mellan mottagningen av nya satellitplatsdata. Det finns även i detta fall olika algoritmer som kan implementeras men efter mycket forskning och enligt [20] kom jag fram till att använda en variation av ett Kalman-filter, en *Master Slave Adaptive Unscented Kalman filter* (MS AUKF). MS AUKF-filtret är anpassat till olinjära system, som det vi har i tröghetsnavigatorn, och kan även anpassa sig till förändringar i systemet snabbare än andra filter.

För att se att systemen fungerar kommer två olika experiment att utföras. Det första kontrollerar om tröghetsnavigatorn fungerar som den ska och jämför sedan den med det mycket dyrare Applanix APX-15 UAV-systemet. Det andra experimentet undersöker möjligheten att förbättra noggrannheten av punktmolnet som bildas av LiDAR 3d-skannaren. Det första experimentet utfördes genom att ha både mitt och det dyrare systemet samtidigt monterade på en cykel medan man cyklade längs en förutbestämd väg. Tröghetsnavigatorerna sparade de data som beräknades under cykelturen så att man senare kunde rita platsdata på en karta. Båda systemen använder en Odroid C2 men vårt system använde den för att beräkna datafusionen medan Applanix APX-15 UAV-systemet använder den bara för att spara platsdata.

Det andra experimentet utfördes genom att använda tröghetsnavigatorn medan LiDAR-systemet utförde en skanning. Data från båda systemen sparades och bearbetades sedan på en annan dator som utförde datafusionen. Slutprodukten från fusionen var ett punktmoln som även innehöll platsdata, det vill säga ett georefererat punktmoln. Experimentet utfördes inom det användningsområde som LiDAR 3d-

skannaren kommer att användas till och objektet som skannades under experimentet var ett egnahemshus.

Resultat

Det första experimentet visade att systemet fungerade och den resulterade noggrannheten från min tröghetsnavigator var bra med tanke på priset av systemet. Under experimentet bytte jag dessutom antennerna mellan satellitmottagarna för att se hur mycket antennerna påverkade systemen. Resultatet var att antennerna påverkade mycket, såpass mycket att mitt systems noggrannhet var nästan lika bra som noggrannheten av det dyrare systemet, Applanix APX-15 UAV, som är 18 gånger dyrare än mitt system. Bilder från experimentet hittas både i texten och i bilaga 1 nedan. I bild 10 ser man en karta på den planerade rutten som utfördes med cykel, och i bilderna 16–19 ser man båda systemens platsdata på samma karta. Platsdata från båda systemen sparades under resans gång och ritades på kartan med hjälp av mjukvara då experimentet var slut.

Slutsatsen från experimentet är att det är möjligt att uppnå ett mycket noggrant system med relativt billiga komponenter. Det lönar sig dock att investera i en dyrare antenn än de andra komponenterna, det fanns en stor prestandaskillnad mellan den billiga och den dyra antennen. Mitt system har dock sina nackdelar, det är större än APX-15-systemet då alla komponenter som användes i mitt system har sina egna kretskort. I mitt system fanns det också vissa programmeringsfel som jag inte har hunnit lösa, till exempel har tröghetsmätarna en tendens att inte skicka data då systemet startar och systemet måste startas om.

Resultaten från det andra experimentet visade att platsdata hade sparats och använts i SLAM-algoritmen för att hjälpa till med att bygga punktmolnet, men punktmolnets noggrannhet hade inte förbättrats märkbart. I bilderna 21–24 kan man jämföra punktmolnen som byggdes av LiDAR-systemet.

Orsaken till att punktmolnet inte förbättrades är troligen användningsområdet som Brighthouse Intelligence LiDAR-systemet används till, skannandet av större objekt. SLAM-algoritmen kopplar ihop enskilda punktmoln genom att identifiera särdrag i ett moln med ett annat. Då man skannar objekt såsom hus kommer det att finnas

tillräckligt många särdrag som SLAM-algoritmen kan använda för att bygga punktmolnet. Ifall vi hade installerat systemen på en bil och kört en längre sträcka, exempelvis runt ett mera öppet område, skulle vi ha fått ett annat resultat då vi byggde punktmoln med platsdata respektive utan platsdata. Detta har bevisats av andra som har experimenterat med att implementera platsdata i ett större punktmoln [36].

Punktmoln och tröghetsnavigatorer är intressanta ämnen som ännu kan utvecklas mycket. Det finns många dyrare tröghetsmätare som man kan experimentera med och SLAM-algoritmerna som bygger punktmolnen utvecklas kontinuerligt. Framtida experiment kommer att inkludera monterande av LiDAR-system på en bil eller båt för att få mera information om hur mycket tröghetsnavigatorn kan förbättra skannandet av större och mera öppna områden.

References

- [1] Lawrence A, *Modern Inertial Technology, Navigation, Guidance and Control*, Second Edition. New York: Springer-Verlag, 2001.
- [2] G. Dudek and M. Jenkin, “Inertial Sensing, GPS and Odometry”, in *Springer Handbook of Robotics*, B. Siciliano, O. Khatib, Eds. Heidelberg, Berlin: Springer-Verlag, 2016, pp- 737 – 751.
- [3] T. M. Adams and R. A. Layton, *Introductory MEMS, Fabrication and Applications*. Boston, MA: Springer, 2010.
- [4] M. Andrejašič, *MEMS Accelerometers*, Marec: University of Ljubljana, 2008.
- [5] A. L. Herrera-May, L. A. Aguilera-Cortés, P. J. García-Ramírez and E. Manjarrez, “Resonant Magnetic Field Sensors Based On MEMS Technology,” *Sensors*, 9, pp. 7785-7813, 2009.
- [6] J. B. Bancroft and G. Lachapelle, “Data Fusion Algorithms for Multiple Inertial Measurement Units,” *Sensors*, 11, pp. 6771-6798, 2011.
- [7] J. B. Bancroft, “Multiple IMU Integration for Vehicular Navigation,” presented at ION GNSS, Savannah, Gam 2009.
- [8] R. G. Valenti, I. Dryanovski and J. Xiao, “Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs,” *Sensors*, 15, pp. 19302-19330, 2015.
- [9] S. O. H. Madgwick, “An efficient orientation filter for inertial and inertial/magnetic sensor arrays,” internal report, 2010.
- [10] K. J. Waldron and J. Schmiedeler, “Kinematics”, in *Springer Handbook of Robotics*, B. Siciliano, O. Khatib, Eds. Heidelberg, Berlin: Springer-Verlag, 2016, pp. 11-35.
- [11] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, “Averaging Quaternions”, [Online]. Available: http://www.acsu.buffalo.edu/~johnc/ave_quat07.pdf. [Accessed Aug. 16, 2018].
- [12] Official U.S. government information about the Global Positioning System (GPS) and related topics, “Space Segment,” *National Coordination Office for Space-Based*

Positioning, Navigation, and Timing. [Online]. Available: <https://www.gps.gov/systems/gps/space/#orbits>. [Accessed Feb. 19, 2018].

[13] Information and analysis center for positioning, navigation and timing, “GLONASS History,” *Information and analysis center for positioning, navigation and timing*, [Online]. Available: <https://www.glonass-iac.ru/en/guide/index.php>. [Accessed Feb. 19, 2018].

[14] European Commission, “The history of Galileo,” *European Commission*, [Online]. Available: https://ec.europa.eu/growth/sectors/space/galileo/history_en. [Accessed Feb. 19, 2018].

[15] European Global Navigation Satellite Systems Agency, “The Results are In: Galileo Increases the Accuracy of Location Based Services,” *European Global Navigation Satellite Systems Agency*, [Online]. Available: <https://www.gsa.europa.eu/newsroom/news/results-are-galileo-increases-accuracy-location-based-services>. [Accessed Feb. 19, 2018].

[16] European Space Agency, “GALILEO’S CLOCKS,” *European Space Agency*, [Online]. Available: http://www.esa.int/Our_Activities/Navigation/Galileo/Galileo_s_clocks. [Accessed Feb. 19, 2018].

[17] H. Durrant-Whyte and T. C. Henderson, “Multisensor Data Fusion”, in *Springer Handbook of Robotics*, B. Siciliano, O. Khatib, Eds. Heidelberg, Berlin: Springer-Verlag, 2016, pp- 867 – 892.

[18] R. Capua and A. Bottaro, “Implementation of the Unscented Kalman Filter and a simple Augmentation System for GNSS SDR receivers,” Proceedings of the 25th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2012), Nashville, TN, September 2012, pp. 2398-2407.

[19] E. A. Wan and R. van der Merwe, “The Unscented Kalman Filter for Nonlinear Estimation,” Oregon Graduate Institute of Science & Technology, 2000.

[20] J. Han, Q. Song and Y. He, ”Adaptive Unscented Kalman Filter and Its Applications in Nonlinear Control,” in *Nonlinear Control, Kalman Filter Recent Advances and Applications*, V. M. Moreno and A. Pigazo, Intech, 2009.

- [21] Basic Air Data, “Inertial Measurement Unit Placement,” *Basic Air Data*, [Online]. Available: <https://www.basicairdata.eu/knowledge-center/compensation/inertial-measurement-unit-placement/>. [Accessed Feb. 26, 2018].
- [22] K. Konolige and A. Nüchter, “Range Sensing”, in *Springer Handbook of Robotics*, B. Siciliano, O. Khatib, Eds. Heidelberg, Berlin: Springer-Verlag, 2016, pp- 783 – 810.
- [23] Velodyne LiDAR, “Ultra Puck™,” *Velodyne LiDAR*, [Online]. Available: <http://velodynelidar.com/vlp-32c.html>. [Accessed Feb. 27, 2018].
- [24] Autodesk knowledge network, “About Point Clouds and LiDAR Data,” *Autodesk*, [Online]. Available: <https://knowledge.autodesk.com/support/autocad-map-3d/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/MAP3D-Use/files/GUID-7C7DD8A7-B561-45B0-A803-852E0A667F3C-htm.html>. [Accessed Feb. 27, 2018].
- [25] Point Cloud Library, “What is PCL,” *Point Cloud Library*, [Online]. Available: <http://pointclouds.org/about/>. [Accessed Feb. 27, 2018].
- [26] V. Magnier, D. Gruyer and J. Godelle, “Automotive LiDAR objects Detection and Classification Algorithm Using the Belief Theory,” In Proc. IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 746-751.
- [27] J. Zhang and S. Singh, “LOAM: Lidar Odometry and Mapping in Real-time,” *Robotics: Science and Systems*, Berkley, CA, USA, 2014.
- [28] W. Hess, D. Kohler, H. Rapp and D. Andor, ”Real-Time Loop Closure in 2D LIDAR SLAM,” in Proc. IEEE International Conference on Robotics and Automation (ICRA) 2016, pp. 1271-1278.
- [29] C. Stachniss, J. J. Leonard and S. Thrun, “Simultaneous Localization and Mapping”, in *Springer Handbook of Robotics*, B. Siciliano, O. Khatib, Eds. Heidelberg, Berlin: Springer-Verlag, 2016, pp- 1153 – 1175.
- [30] ROS, “History,” *Open Source Robotics Foundation*, [Online]. Available: <http://www.ros.org/history/> . [Accessed Jun. 14, 2018]
- [31] ROS, “About ROS,” *Open Source Robotics Foundation*, [Online]. Available: <http://www.ros.org/about-ros/>. [Accessed Jun. 14, 2018]

- [32] ROS, “ROS Introduction,” *Open Source Robotics Foundation*, [Online]. Available: <http://wiki.ros.org/ROS/Introduction>. [Accessed Jun. 14, 2018]
- [33] Trimble, “Version 3 Single Board GNSS-Inertial Solution,” APX-15 UAV datasheet, 2017.
- [34] ROS wiki, “imu_complementary_filter,” *Open Source Robotics Foundation*, [Online]. Available: http://wiki.ros.org/imu_complementary_filter. [Accessed Feb. 21, 2018]
- [35] ROS wiki, “robot_localization,” *Open Source Robotics Foundation*, [Online]. Available: http://wiki.ros.org/robot_localization. [Accessed Feb. 21, 2018]
- [36] Github, “Google cartographer NavSatFix messages”, *github.com* [Online]. Available: <https://github.com/googlecartographer/cartographer/issues/1014> [Accessed Sep. 19, 2018]
- [37] COMMISSION OF THE EUROPEAN COMMUNITIES, “*Galileo, Involving Europe in a New Generation of Satellite Navigation Services*,” in COMMUNICATION FROM THE COMMISSION, Luxemburg: Office for Official Publications of the European Communities 1999.
- [38] Pololu Corporation, “*MinIMU-9 v5 Gyro, Accelerometer, and Compass (LSM6DS33 and LIS3MDL Carrier)*,” *pololu.com*, [Online]. Available: <https://www.pololu.com/product/2738> [Accessed Nov. 20, 2017]
- [39] CloudCompare, “3D point cloud and mesh processing software Open Source Project,” *cloudcompare.org*, [Online]. Available: <http://www.cloudcompare.org/> [Accessed Oct. 2, 2018]
- [40] R. Mahony, R. W. Beard and V. Kumar, “Modeling and Control of Aerial Robots,” in *Springer Handbook of Robotics*, B. Siciliano, O. Khatib, Eds. Heidelberg, Berlin: Springer-Verlag, 2016, pp- 1307 - 1334.
- [41] *National Geospatial-Intelligence Agency*, “World Geodetic System 1984,” G1670.

[42] T. G. Gainer and S. Hoffman, *Summary of transformation equations and equations of motion used in free-flight and wind-tunnel data reduction and analysis*, Washington, D.C.: National aeronautics and space administration, 1972.

[43] Tallysman, "A Tallysman Accutenna, TW3870/TW3872 GPS L1/L2 + GLONASS G1/G2 + BeiDou B1 + Galileo E1," TW3870 datasheet Rev4.3, 2010.

[44] Aliexpress, "Car GPS Antenna GPS receiver Car DVD Navigation Night Vision Camera Car DVR GPS Active Remote Antenna Aerial Adapter Connector," [Online] Available: https://www.aliexpress.com/item/Car-GPS-Antenna-GPS-receiver-Car-DVD-Navigation-Night-Vision-Camera-Car-DVR-GPS-Active-Remote/32788800179.html?spm=2114.search0104.3.43.435b6c59VFSgjx&ws_ab_test=searchweb0_0,searchweb201602_2_10065_10068_319_317_10696_5728811_10084_453_454_10083_10618_10304_10307_10820_10821_537_10302_536_5733215_5733315_328_10059_10884_5731015_10887_5733115_100031_5733415_321_322_10103_5733515_5733615,searchweb201603_1,ppcSwitch_0&algo_expid=b73cc6a7-5b9c-4f27-a309-5ea064ef168e-6&algo_pvid=b73cc6a7-5b9c-4f27-a309-5ea064ef168e [Accessed Nov. 8, 2018].

[45] GitHub, "C++ 11 NMEA Parser and GPS Interface," ckgt, Available: <https://github.com/ckgt/NemaTode>, [Accessed Oct. 30, 2017]