UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Siim Plangi

Real-time Localisation and Tracking System for Navigation Based on Mobile Multi-sensor Fusion

Master's Thesis (30 ECTS)

Supervisor: Amnir Hadachi, PhD
Supervisor: Artjom Lind, MSc

Tartu 2018

Real-time Localisation and Tracking System for Navigation Based on Mobile Multi-sensor Fusion

Abstract:
With the rise of the smartphone, new research opportunities have emerged. With a wide array of sensors that are available in today's smartphones, the research possibilities are endless. In this work, we present a new algorithm that can track and localise a vehicle in real-time using the GPS, accelerometer and gyroscope data streams from an Android OS smartphone. The resulting algorithm can respond to speed changes, and the car turns in real-time without any info from the GPS. This means that the algorithm can estimate the vehicle position if the GPS data stream is unavailable for unknown amounts of time. Results are promising and show that the algorithm performs well both in accuracy and real-time responsiveness. Even without 30 seconds of GPS info, the algorithm is able to estimate the vehicle location with an average accuracy of 25 meters.

Keywords:
Multisensor Fusion, Localisation, Tracking, Android, GPS, Kalman Filter

CERCS:
P170, Computer science, numerical analysis, systems, control

Mobiilsete sensorite integratsioonil põhinev reaalajaline lokaliseerimine ja jälgimissüsteem

Lühikokkuvõte:
Nutitelefoni tõusuga ja nendesse paigaldatud anduritega on tekkinud lõputult teaduslikke uurimisvõimalusi, ilma kallist riistvara omamata. Selles töös tutvustame uut algoritmi, mis võimaldab jälgida ja lokaliseerida sõidukit reaalajas, kasutades Android OS nutitelefoni GPS-i, kiirendusmõõturi ja güroskoobi andmevoogusid. Loodud algoritm võib reageerida kiiruse muutustele ja auto pööretele reaalajas ilma GPS-i sisendita. See tähendab, et algoritm saab hinnata sõiduki positsiooni, kui GPS andmevoog ei ole teadmata ajahulgal saadaval. Tulemused on paljutõotavad ja näitavad, et algoritm toimib hästi nii täpsuse kui ka reaalajas reageerimisega. Isegi ilma GPS infota 30 sekundit jooksul suudab algoritm hinnata sõiduki asukohta 25 meetrilise keskmise täpsusega.

Võtmesõnad:
Sensorite integratsioon, lokaliseerimine, jälgimine, Android, GPS, Kalmani filter

CERCS:
P170, Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1 Introduction

Nowadays, Modern smartphones are not just for browsing the web, making phone calls and chatting with friends anymore. The rise of smartphones has, in addition to changing our everyday lives, also provided the scientific community with a new opportunity to use the smartphone as a scientific tool with the connection to the Internet and with its many embedded sensors. The scientific world today uses the smartphone for various applications including localisation and tracking. The embedded sensors, which include the GPS, gyroscope and accelerometer, are ideal and more than enough for tracking the position of the smartphone accurately in real time.

The relevance of real-time tracking and localisation is increasing with every year. Many fields can be named, that make use of knowing the exact location of the user or device. For example localisation and tracking is used in parking systems [21], racing and rally [10], and fleet management[25]. Each of these fields needs the data in real - time, e.g. with minimal delay from the current time. Also, having an accurate location is as important. Failing to produce real-time and accurate location data in these fields could result in a potential financial loss, material damage or loss of life. Moreover, with autonomous vehicles and other autonomous technology on the horizon, the awareness of accurate, current location at real-time is essential for these technologies.

1.1 Objectives and limitations

This thesis proposes a software to track and localise vehicle movement in real time using only the smartphones embedded sensors. The objective is to demonstrate software that uses the embedded sensors of a smartphone, namely GPS, gyroscope and accelerometer, to produce an accurate location estimation in real-time, even if GPS signal is lost for periods of time. The response time for location request must also be adequate. The software quality was assessed from multiple viewpoints; the following targets were set:

- The software must respond to vehicle turns
- The software must respond to vehicle accelerations and decelerations
- The software must be able to work in real time and produce at least 50 estimations per second.
- The software must be able to continue providing estimations when GPS updates are lost.

Limitations There are also some limitations that can affect the proper functioning of the software. For example, the positioning of the smartphone inside the vehicle. Firstly,

the smartphone must be fixed inside the car, for the smartphone to pick up inputs only from vehicle movements, not from smartphone moving around in the vehicle. Secondly, the assumption is made that the smartphone is set inside the car with the back of the phone directed to the vehicle forward moving direction. This assumption was needed to separate the turn forces from the acceleration forces to the device's accelerometer.

1.2 Contribution

In this thesis, the work can be divided into three parts:

- Firstly the research phase, where the selection of methods and filters were made. Many filters were considered to be used in our system, but three were chosen: Kalman Filter, Moving average Filter and the Simple moving average filter.
- Secondly the system design. The system itself consists of different modules, and the modules consist of different algorithms, which help to alter the data stream to produce the wanted result. Each of these modules uses a custom developed mechanism and algorithms inspired by the filters as mentioned earlier. In total seven different size modules were developed. The modules are Gravity, Acceleration, Angular speed, Speed fusion, Heading fusion, Tracking and localisation fusion and Map-matching module.
- Thirdly, the implementation of the Android application itself where to put this technology to the test. The Android app composes of 3180 Java SLOC (source lines of code) and 301 XML descriptor SLOC. The application includes a map view, where the results can be viewed in real time, instead of just logging the results into memory. The map data also is queried over the network in real time, so the application itself is usable anywhere around the world with internet connectivity without the application itself including any static map data.

1.3 Road map

There are four major sections in this thesis, and they are as follows:

- State of the art - In this section relevant state of the art technologies are researched and discussed. Three different viewpoints are taken. Firstly, Real-Time systems are discussed. Secondly, state of the art Localisation and tracking methods are examined. Finally, the Location prediction methods are researched.
- System design and architecture - Secondly, the proposed system architecture is discussed. The choice of platforms used in this project is reasoned, input data indicated, and the overall system operation explained.

- Modules and Components - In this section, the main working procedure of the system is isolated into modules. These modules are then thoroughly examined, and their processes are reasoned.
- Results and Discussion - In this last main section, the results of system testing are displayed. The results are then discussed, and explanations regarding the results are provided.

This is the end of the introduction part of the Thesis. In the next section, the current state of the art systems that are relevant to this work are discussed.

2 State of the art

In this section, the state of the art algorithms and terminology are discussed. The chapter is divided into three subsections, which addresses Real-Time systems, Localisation and tracking and Location prediction.

2.1 Real Time systems

Real-time systems can be found all around us. From our smartphones to our cars, from our television sets to the electricity providing power plants. Real-Time systems can be defined in the following way: A system in which the correct running behaviour is determined by the logical results of the computations and the physical timeframe when the results are provided. An example of a non-correct behaviour of real-time systems would be if a aeroplanes autopilot would respond with control outputs to a set of inputs (airspeed, wing configurations, altitude, heading, rotation, tilt etc.) with an unexpected significant delay. The results could be catastrophic, e.g. the plane could crash [7].

Timing constraints As Real-time systems have timing constraints then there must be consequences when this deadline is not met. Depending on the implications of missing the real-time deadline systems can be divided into two subgroups.

- Hard real-time systems - When the implication of missing a set deadline is catastrophic or fatal to the process, e.g. loss of life, significant financial loss, non-usable result etc., then the real-time system is defined as Hard real-time systems.
- Soft real-time systems - When the implication of missing a deadline is not catastrophic or non-fatal to the process, e.g. slight human discomfort, non-significant processing delays, then the real-time system is defined as a Soft real-time system.

As real-time systems may compose of multiple real-time subsystems, then the system, which is composed of only soft real-time subsystems is a soft real-time system as well. On the other hand, if a system has key hard real-time subsystems, then the parent system is a hard real-time system as well.

2.1.1 Soft real-time systems

Soft real-time systems fill the timing constraints on best-effort principle, and occasional deadline misses may occur. These deadline misses are tolerable as they do not cause a catastrophic event. It must be noted that even though missing the deadline does not result in a catastrophic event, then the quality of the service may be degraded and thus this is still an unwanted situation. Some examples of soft real-time applications and their implications of missing deadlines were hypothesised.

- Vehicle navigation system - A automobile navigation device is a real-time system because it responds to GPS inputs in a certain timeframe with routing directions. If this system would be unresponsive, then the implication would be that the user might miss the next turn, which is non-catastrophic and thus falls into the realm of soft real-time systems.
- Smartphone camera system - Smartphone cameras task is to take pictures and save them onto a storage medium upon command from the user within the manufacturer proposed timeframe. If the task is not completed within a timeframe, then the result might be a frustrated user.
- Coffee machine - Coffee machine makes coffee on the command of a user. The implications of not delivering coffee quickly enough are not catastrophic.
- Gaming console - A gaming console is a real-time system as user inputs commands via a controller and the gaming console must output the respective view in response to the command. If there is a slight delay or the video stream is "lagging" then the situation is not desirable but not catastrophic and thus has the characteristics of a soft-realtime system.

2.1.2 Hard real-time systems

A missed deadline for head real-time systems is completely unacceptable because it might result in harm to a human being or significant financial loss. Therefore for hard real-time systems, both the correctness of the result and execution completion before the deadline must be guaranteed. In some cases, the timing aspect of a real-time system might be even more important, as the non-complete results can still be used to perform actions or present to the user for an acceptable outcome. Again, some example hard real-time systems and implications of them missing their deadlines were hypothesised.

- Autonomous vehicle object detection system - This system would be responsible for detecting surrounding objects around the vehicle. This system would be a quite high rate and missing deadlines for object detection because it could cause a crash of the vehicle which may result in human harm and financial loss.
- Vehicle anti-skid system - Various anti-skid systems are installed to vehicles to restrict the user losing control over the vehicle. This system would analyse various parameters of the vehicle up to hundreds of times per second and thus must be highly responsive. If an uncontrollable event would be unnoticed or noticed late, then the result can be an accident that is again a catastrophic event.
- Automatic bitcoin trader - This system would download the bitcoin financial data multiple times per minute, make the analysis of the acquired data and decide on

the action. If this system would fail or become unresponsive, then the result might be a financial loss for the user.

2.2 Localisation and tracking

Localisation and tracking have come a long way since the public adopted the smartphones. Nowadays there are various ways to achieve localisation of a device because there are so many sensors that can be exploited for this. The most obvious sensor to use for localisation and tracking would be the GPS that most smartphones and various other devices include. GPS is an excellent device to localise and track a device as its purpose-built for this and can give a good location estimate within few meters of reality. Unfortunately, GPS is not perfect and needs unobstructed skies to work. Therefore it becomes almost useless when tried to use indoors or underground.

Indoors localisation and tracking is a major research area right now as it has a lot of commercial interest. For example, while shopping in a supermarket or mall, the user of the indoor location-aware application can benefit from knowing the exact position within the building and get clear directions to the searched location. It can provide major benefits for the business owner as well from customer satisfaction to various statistics like moving patterns around the building. Another area where indoor localisation and tracking is essential is security. The movement and actions of people must be monitored to prevent theft or potential harm to other humans. There are various approaches to achieve indoor localisation and tracking. The following represents some articles and different methods to achieve localisation and tracking.

2.2.1 GPS localisation and tracking

GPS is a technology that uses the signals sent from earth's satellites and then triangulates the position on earth using the relative distances from the receiver device to the satellites. Results from GPS are not perfect and can be disrupted by anything that blocks the line of sight to the sky, like high buildings, being indoors or driving through tunnels. Therefore there exist many methods to improve the results provided by GPS, like installing more expensive and sophisticated GPS receiver hardware or try to improve the results with software or by fusing together data from other sensors. The latter is a popular research area. State of the art GPS localisation and tracking methods can be divided into two subcategories. Firstly there is the group that focuses on building on algorithms that can fix GPS errors. The second group uses other sensors to improve the position results in areas where GPS fails or provides poor accuracy results.

GPS data enhancement by error fixing: Fixing GPS errors can be achieved by applying some filter to the GPS data. An Example of an error fixing method is the strong tracking filter (STF) which is applied to the GPS data [27]. Strong Tracking filter has the characteristics to give good results even if the precision of the system is unknown. The algorithm utilises adaptive fading factor to create a gain matrix, which gives either more value to the predicted position or the measurement. The results show that the filter is robust and able to outperform Kalman Filter (KF) if there is no error data on the GPS stream, the initial value is erroneous, or the state of the system changes unexpectedly.

GPS data enhancement by sensor fusion: Methods based on fusion, use other available sensors to enhance the position estimation accuracy and make it possible to predict location before the next GPS update. Sensor fusion is especially effective when the location is needed in real-time as GPS position updates come at a limited rate. Getting high-quality location information is essential for future developments of autonomous vehicles. Such case has been studied in article [13], where GPS sensor is fused with in-vehicle sensors as Wheel speed sensor, Steering angle sensor and Yaw rate sensor. As modern standards require that each vehicle has to be equipped with stability control systems and many vehicles today already include in-vehicle built navigation systems, then all of these sensors are already included inside the car. The proposed algorithm uses Extended Kalman Filter for sensor fusion. This article puts a lot of emphasis on the tyre slip conditions. Tyre slip occurs when a lot of force is put on the tyre like driving with high speeds or fast cornering situations, such as driving on highways or mountain roads. Two different models have been proposed for these situations, firstly a standard kinematic model, where zero tyre slip is assumed. Secondly a dynamic model, where the slip of tyres has been accounted for dynamically. It is assumed that the lateral force acting on a tyre is proportional to tyre slip angle. The kinematic model is very accurate on low tyre slip conditions but breaks down on high slip conditions as opposed by the dynamic model. Therefore the algorithm can adapt to current driving conditions, for example, if the vehicle is moving at high speed or under high lateral force situations, a dynamic vehicle turning model is used. Only visual representations of the accuracy of this algorithm are provided, but the visuals show the algorithm can estimate the vehicle position to a high degree.

2.2.2 Inertial sensors

Inertial sensors can successfully be exploited to track and localise when the starting position of the subject is known. In [22] the four types of inertial sensors were attached to a shoe of a pedestrian and the performance of these sensors were evaluated. The four sensors were the acceleration moving variance detector, the acceleration magnitude detector, the angular rate energy detector aka gyroscope, and a novel generalised likelihood

ratio test detector. The end result were promising and showed that they were able to estimate the pedestrian position within 0.14% of travelled distance with combining the results from the likelihood ratio test detector and angular rate energy detector. Inertial sensors are also often used in combination with other technologies to estimate location. For example in [30] the mobile phones inertial sensors are used in combination with Wifi and iBeacon. The method was then put to a test by attaching the sensors to a user and a predetermined route was travelled. The result shows that with the combination of all three technologies improve the results using the inertial sensor alone from a mean error of 2.732m to 0.594m. The main drawback that both articles mention is that using inertial sensors alone is the accumulation of errors over time. Thus a large deviation of the real and predicted position can occur if the test has run a significant amount of time. Therefore most applications use the inertial sensors in combination with other sensors.

2.2.3 Visual aid localisation

Visual aid localisation is the most natural way for humans to assess their location. We use the visual references to navigate in our world. This method is a popular research area especially when developing Humanoid robots. Humanoid robots are kind of robots that visually represent humans body and try to mimic their actions. For example there exists a football league called "RoboCup" that uses humanoid robots to play football against other robots. There are many articles that represent algorithms to detect various objects during a football match. In [29] a visual self-localisation method is described. The algorithms purpose is to detect the position of the robot in the football field using digital images from the cameras and preset reference points on the field like the goals and beacons to produce a coordinate and the angle that the robot is placed on the field. The distances to these preset items is determined using monocular vision and then triangulation is used. The results show, that when the reference points are well detected the positioning error is around 1 cm, but when some problems arise with reference points detection then the error can turn up to 40 cm.

Another field where visual aid localisation and tracking is used is CCTV (closed-circuit television). It is primarily used for manual security and surveillance, but can also be used for statistical analysis and automatic identification if the footage is processed and objects are tracked and localised. As most modern cities use traffic cameras, then this network of cameras can be used for example to model traffic and make predictions of it. Such work has been done in [12] where there is a method described to automatically calculate traffic volume and speed. This has been achieved by pixel pattern analysis. Firstly the lanes of road are detected from the video feed and then when the specific lane pixel brightness change, then it can be assumed that there is a vehicle passing by.
The drawbacks of using Visual aid localisation and tracking are obvious. Firstly, object detection and tracking is a difficult problem to solve, especially when abstraction of an

object is needed. Therefore a lot of processing power is needed. Secondly, objects that are tracked must be in sight of the camera system. When the object leaves the line of sight, then the object localisation and tracking is obsolete.

2.2.4 Indoor localisation methods

Indoor localisation is a difficult problem and is not perfected yet, although more than a decade of research time has put into it. Microsoft holds a yearly competition for Indoor localisation prototypes, with more than 20 yearly ambitious projects [16]. Although there are prototypes that can achieve sub-meter accuracies, the overhead of equipment installation for these methods is high, impractical and often expensive. When the overhead of extra equipment is reduced by using only infrastructure free methods to detect localisation indoors, then besides the drop in accuracy, the overhead of mapping remains, and the accuracy is highly dependent on changes in the environment. No method yet exists that works indoors as well as GPS does outdoors. Two most common methods exist for Indoor Localisation. One of the methods being fingerprinting, which essentially is mapping the whole indoor area with data and then later, when location needs to be tracked, this mapping is compared with current data and the most probabilistic position is chosen. Second method is Triangulation. Triangulation is a method used to detect a point by measuring the distance to known points. A minimum of three points is needed to be known for a effective triangulation to take place. The same method is used with GPS technologies, as the position of the satellites are known and therefore by knowing distances to satellites a position on earth can be determined. Same ideas can be applied indoors as well, with various signals, like WiFi, Bluetooth or Radio. Following is the introduction of some of the most common approaches to indoor localisation.

Infrastructure independent methods: 802.11 protocol e.g. WiFi is often exploited to track the position of a device indoors, because its low cost and has high availability indoors. The low cost and high availability mean that localisation can usually be achieved without employing extra hardware indoors. The most prevailing way to use WiFi signals is using RSSI (Received Signal Strength Indicator) fingerprinting [16]. This method compares the WiFi signal strengths from various access points to previous mappings of positions and WiFi signal strengths and chooses the best match from the past data. While this provides a method to achieve localisation without employing extra hardware, it is highly dependant on mapping quality Also hardware differences for mapping and localisation causes inaccuracies [3]. Another way to exploit the WiFi signal to determine the position of a device supporting IEEE 802.11g signal is to use multiple mutually synchronised 802.11g receivers to obtain the Time of Arrival (TOA) of the signal [17]. From the TOA a distance between the transmitter, e.g. the target device and a receiver

e.g. fixed positioned router can be determined. Although better results than RSSI fingerprinting can be achieved, this method is limited with line of sight assumptions. When some obstacles are introduced the error of this method increases significantly.

Infrastructure dependent methods: Infrastructure dependent systems can use various technologies to achieve indoor localisation like Bluetooth beacons, LIDAR, Ultra Wideband radio or Sound. All of these technologies have different cost and accuracy. One inexpensive infrastructure dependent way to achieve indoor localisation is using Bluetooth beacons. Similarly to using WiFi signals, Bluetooth beacons exploit the same ideas: RSSI fingerprinting and triangulation. In [18] Bluetooth beacons are successfully used to achieve Indoor Localisation. A particle filter is used based on the floor map and the proximity of BLE beacons are used. According to authors an average error of 0.999m is achieved, which is better than similar algorithms.

One of the most advanced and expensive technology used in Indoor localisation is using LIDAR. Lidar is a technology which creates a high resolution point distance cloud of its surroundings. In [20] the building is first mapped with LIDAR, and a high definition point cloud is created of the building. When localisation is taking place, then this point cloud is matched to the current data. As Lidar creates a very high resolution image and analysing all in real time that takes a lot of processing power, then some selection of point data from LIDAR must be taken. Therefore the authors employ feature detection to select out only the most feature some points from the cloud. The results show that the location estimation error is on average about 20cm from the reality, which is a good result.

2.3 Location Prediction

Two most prominent ways to achieve location predictions is Map-Matching, which predicts the probable location using map data in real-time. Another group of location predictions algorithms use historical data and make assumptions based on that. Both of this aforementioned methods are used in different situations. Whilst Map-matching is usually used to correct data and to provide a smoother, more realistic output to the user, then movement predictions are used to predict future actions and movements of users. In this paragraph both of these methods are examined and their advantages and disadvantages are analysed.

Historical movement: Historical movement data of humans can be used to predict the future movements of individuals. This is a powerful knowledge that can be used to help people, save the environment or for monetization purpose. An example of helping the

environment would be turning on the heat at home only when someone is heading home from family. Or ride-sharing purposes, getting multiple humans on the same vehicle, just because you know where they are heading. The possibilities are limitless. There are multiple algorithms that can achieve such predictions in the literature. Authors of [28] propose an algorithm that can predict the movement of individual with an accuracy of 85.5% from historical data. Location data is uploaded from the recording device to the server and predictions of next movements are displayed on a client-side browser. The idea behind the prediction algorithm is using a simplistic probability model, which is defined as the probability of user moving to the next region r_{next} when the input is the user being in region r. This however does not take into account the times of travel, e.g. when its 8 o'clock the user is probably going from home to work instead of work to grocery store. That's why they introduce the time variable into their probabilistic model. Therefore the predictions become more accurate, the more data is uploaded to the server by the user, as user movements are usually time based.

Map matching: As most of human travels happen on predefined paths then this can be successfully used to predict the future movements of people. Combining this with a probability model of paths then it can be a powerful short-term prediction method. The authors of [24] describe an online map matching algorithm that is able to predict future movements of users. They use a probabilistic model to predict the route of the users by creating a road graph and then assigning probabilities to each branching. This probabilities are able to change when more data is added and the model is further trained. The accuracy of their algorithm depends on the sampling rate of GPS inputs, with accuracy degrading when the sampling is more infrequent. The proposed algorithm achieved an 94% accuracy with trained data model and 92% accuracy with untrained model using 30s GPS sampling rate. The testing was done in Nagakute, Japan, with a total driving time of 2 hours.

3 System design and architecture

In this paragraph, a new method is proposed for location fusion and tracking. The methods design and its architecture are explained in detail in the subparagraph. Firstly, the choice of Android ecosystem and its possibilities are described. Secondly, the OpenStreetMap database and Overpass system are introduced. Thirdly, the overall scheme and architecture of the tracking and localisation system are described.

3.1 Android ecosystem

The proposed method is built and tested on the Android ecosystem. The choice for Android run mobile devices was taken due to many factors.

- Android is by far the most popular operating system worldwide from all the mobile operating systems [17]. About 75% of mobile devices run Android OS as of February 2018, with the second most popular operating system, iOS, used by 20% of devices. The extensive usage of Android reflects in the scientific community, and thus the research can be used directly in other projects as well.
- Android apps are written in JAVA, which is one of the most popular programming languages out there. There are sources that rate it the most programming language, others claim it to be in the TOP 3 [4, 23]. Using JAVA as the programming language makes it more likely, that someone can take advantage of the project in the future. Also, the author had previous experience writing Android apps in Java, which made a choice even more desirable.
- Android testing devices are easy to obtain, and most of these devices have a wide range of available sensors. As the author of this paper has multiple Android devices that can be used for testing purposes, then it made sense to use an Android ecosystem.

3.1.1 Available hardware sensors

Android supports many hardware sensors, which the manufactures can install to the device, for the OS to take advantage of. Also, android provides software sensors, where outputted data is calculated using a combination of other sensors. According to the Android documentation [15], the sensors that are supported and maintained are listed in Table 1. The localisation and tracking method uses a subset of available hardware sensors to provide real-time location info:

- GPS
- Accelerometer

- Gyroscope

The availability of these sensors in a device is a requirement for the localisation and tracking method to work correctly.

Sensor	Type	Description
GPS	Hardware	Measures the device position on earth and other parameters like speed and heading
Accelerometer	Hardware	Measures the acceleration force that is applied to a device on three device axes (x, y, and z). Includes the force of gravity
Ambient temperature	Hardware	Measures the room temperature
Gravity	Software or Hardware	Measures the force of gravity in m/s^2 that is applied to a device on all three physical axes (x, y, z)
Gyroscope	Hardware	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, and z)
Light sensor	Hardware	Measures the ambient light level (illumination) in lx
Linear accelerometer	Software or Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), excluding the force of gravity
Magnetic field	Hardware	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in μ T
Pressure	Hardware	Measures the air pressure
Proximity	Hardware	Measures the distance of an object to the sensor. Typically this is used for determining if the user is holding the phone against the users head.
Relative humidity	Hardware	Measures the relative ambient humidity in percent ($\%$)
Rotation vector	Software or Hardware	Measures the orientation of a device by providing the three elements of the device's rotation vector.

Table 1. Android sensor support

3.2 Open Street Maps and Overpass

OpenStreetMap (OSM) is a community built geographic map database [8], using community provided Ariel images, GPS data and low tech field maps. All the data is submitted by volunteers that can create and update the map data anywhere in the world. This data is open, meaning that everyone can use it for free, with the only requirement being, that OpenStreetMaps are credited. OSM includes roads, buildings, landscapes, points of interests and other data of the world.

As raw OSM data itself is more than 65GB large compressed, then obviously some subset of the map data must be chosen to access only the interesting dataset. Overpass Turbo API is a web-based tool [2] that can return a subset of map data based on a query. This API can be used to query a piece of map data that is interesting to the user. This tool then can return a bounded data object only with interesting data objects. For example one would be interested in only road-map of a city, then Overpass Turbo API can provide such a data. In the application, this API is used to query the surrounding area from the user location perspective, and this data is then cached to be used again. When the user starts to leave the bounding area, by arriving at the edge of the bounds, then another area is queried from the API to ensure a continuous road map data.

3.3 Design and architecture

In the following, the architecture and overall design of the localisation and tracking application are discussed. The big picture composes of three different phases (Figure 1):

- Data gathering phase - hardware sensor outputs are gathered and mapped to a suitable form.
- Location fusion phase - The input data is then filtered and fused into more accurate and higher frame rate data-stream.
- Map Matching phase - The location fusion output data is then constantly matched with map data, and a final localisation estimation is provided.

The subsections onwards go into these phases more deeply, discuss their tasks and how the results are achieved.

3.3.1 Input data

Location data and tracking require some raw input data from the device. The input data for location fusion and tracking is gathered from three different hardware sensors. Following is a list of the required sensors and description of their outputted data:

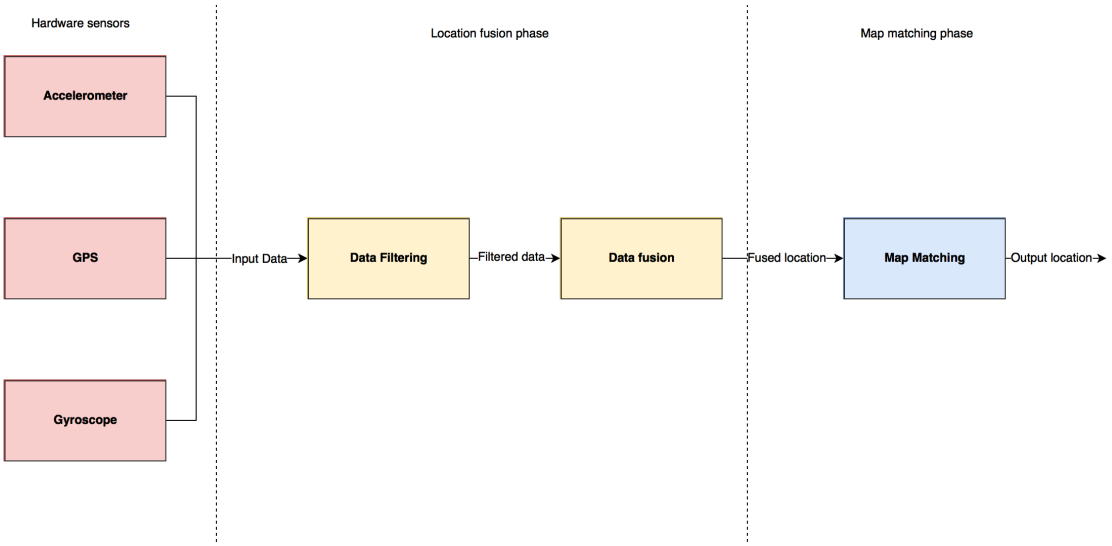


Figure 1. Overall architecture of the algorithm

- GPS - GPS sensor is the primary measurement tool which outputs the devices speed, heading and coordinate. The frame rate of the outputted data on Android devices peaks usually at 1Hz. The low GPS update rate means that this sensor cannot provide enough detail for real-time localisation and tracking and thus needs to be improved.
- Accelerometer - This sensor detects the phones accelerations and decelerations on three different axes (x,y,z) and their strength [Figure 2]. The outputted frequency is device dependent but can exceed 100hz on higher-quality smartphones. Android provides methods to choose the data output frequency. Higher frequency increases noise and vice versa. The sensor is used because of two reasons. Firstly, its used to detect the accelerations and decelerations of the vehicle to estimate the speed. Secondly, its used to identify the devices rotational position in reference to earth.
- Gyroscope - Measures the device's rate of rotation on all three physical axes (x,y,z) [Figure 3] and outputs the data in rad/s per axis. Again the output frequency is device dependent but can exceed 100hz. Android provides a selection of frequency outputs, with higher rate having higher noise and vice versa. The usage of this sensor is needed to observe the vehicles turns.

3.3.2 Fusion phase

Location fusion consists of two subtasks. Firstly the input data is filtered and modified. Secondly, the data is fused together using Kalman Filter to produce accurate real-time

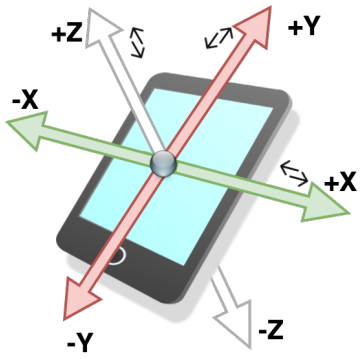


Figure 2. Smartphone's accelerometer measures accelerations on x,y,z axis

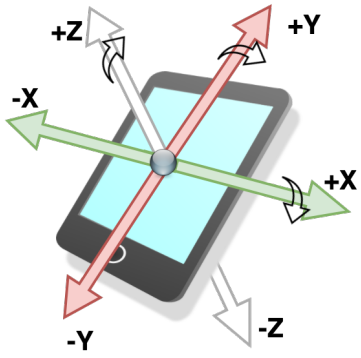


Figure 3. Smartphone's gyroscope measures accelerations on x,y,z axis

location, heading and speed estimations. The filters and fusing methods used in the tracking and localisation algorithm and their short descriptions are:

- Simple moving average filter (SMA) - SMA filter is used to smooth out a noisy data stream and thus can provide values closer to the reality. SMA filter is used on a time series data. A time interval of values are selected from the input data, and the average function is applied. The result is then outputted. In the localisation and tracking algorithm, the simple moving average filter is used to smooth out the accelerometer and gyroscope outputs.

- Exponential moving average (EMA) filter - Similarly to SMA, the EMA filter is used to smooth out a noisy data stream. EMA filter computes its output by taking values from a time series data stream, assign weights to those values and then calculate an average. It is shown that EMA filter can perform better than simple MA filter and can give smoother results than simple MA filter [6]. EMA filter is used in the algorithm context to isolate the gravity from the accelerometer data. The priority is given to earlier data, and an average estimation of data is outputted.

- Kalman filter (KF) - Kalman filter uses a series of measurements over time and the estimated errors of those observations as input and outputs an estimation of a true state. The measurement error distribution must be Gaussian for the Kalman filter to achieve the best performance. Kalman filter is also capable of prediction of future state given the right inputs. In the context of localisation and tracking algorithm, this filter is used to fuse the location data together. Kalman filter has this ability to use data streams from multiple sources, like GPS, accelerometer and gyroscope to produce a fused data output stream. Besides, to ensure its proper functioning effectively, it is needed to have an accurate error estimation for both the measurement inputs and the filtering process itself. This latter is needed since a lower measurement error gives more weight to the measurement data stream and lower process error gives more weight to the filters own predictions.

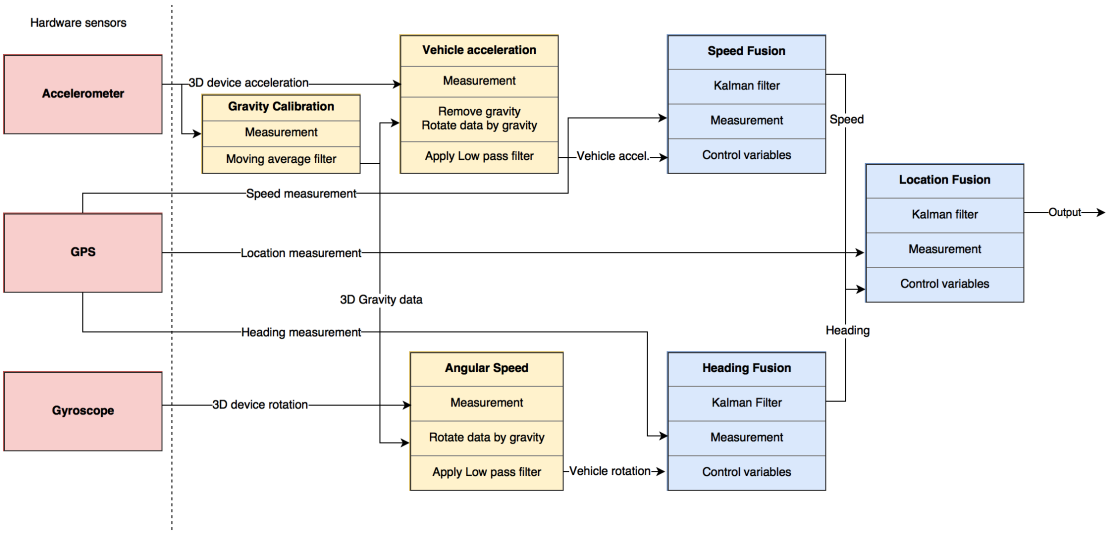


Figure 4. Archidecture of the location fusion phase

Figure 4 illustrates the architecture and design of location fusion phase. The inputs from the hardware sensors are sent to different location fusion modules. These modules are responsible for determining different states of the system:

- Gravity calibration - This module's task is to accurately capture the gravity strength on each smartphone accelerometer (x, y, z) axes (Figure 2). A custom algorithm has been implemented for estimating the gravity value, which differs from the Android standard gravity sensor.

- Vehicle acceleration - Measures the acceleration and deceleration of the vehicle in one direction. It uses the accelerometer for extracting measurements, Simple

moving average filter to smooth out the data and the gravity sensor to refine the output.

- Angular speed - Detects vehicle turns using gyroscope output as its measurements. It also uses the gravity to rectify the data and the low pass filter to smooth out any noise from the gyroscope.
- Speed fusion - this module is in charge of accurate estimations of speed. Kalman filter is used to fuse the GPS speed outputs and vehicle acceleration sensor outputs together.
- Vehicle heading - Keeps track of the bearing of the vehicle. It uses Kalman filter to fuse together the GPS heading and the gyroscope output.
- Location fusion - Uses Kalman filter that takes data from the GPS, Vehicle heading and Speed module to estimate the location of the vehicle.

The output of the phase provides enhanced and high-frequency estimations of location, speed and heading. Each of these modules will be discussed at length in the next Section.

3.3.3 Map matching phase

After the location fusion phase has been completed, then the next task of the algorithm is to map match the location. This is a three-part task (Figure 5). Before map matching can be done, the detailed road maps must be acquired for the current location. This map retrieval process is split into two tasks - caching and download. Instead of creating fixed implementations, the tasks are created as interfaces. Interfaces help to make sure that everyone can create a map retrieval process that is suitable for them. For instance, one could implement map downloading and cache by just providing a static copy of the map on to the device. Another example would be to download the map data dynamically from the network and cache the data onto to device. The latter is also implemented in the algorithm and can be used. The third task is to map match the location and provide the outputs.

Map Data Fetcher: An interface class that has the task to retrieve map data from external sources. Can be implemented to fetch data from the device local storage or various network sources. An implementation is provided, that downloads the map data from Overpass Turbo API. The map data provider asks to fetch map tile. The bounding map tile north, south, east and west coordinates are then inputted into the network query, and the map data is retrieved as a MapTile object. The MapTile object includes all the roads of the surrounding bounding box.

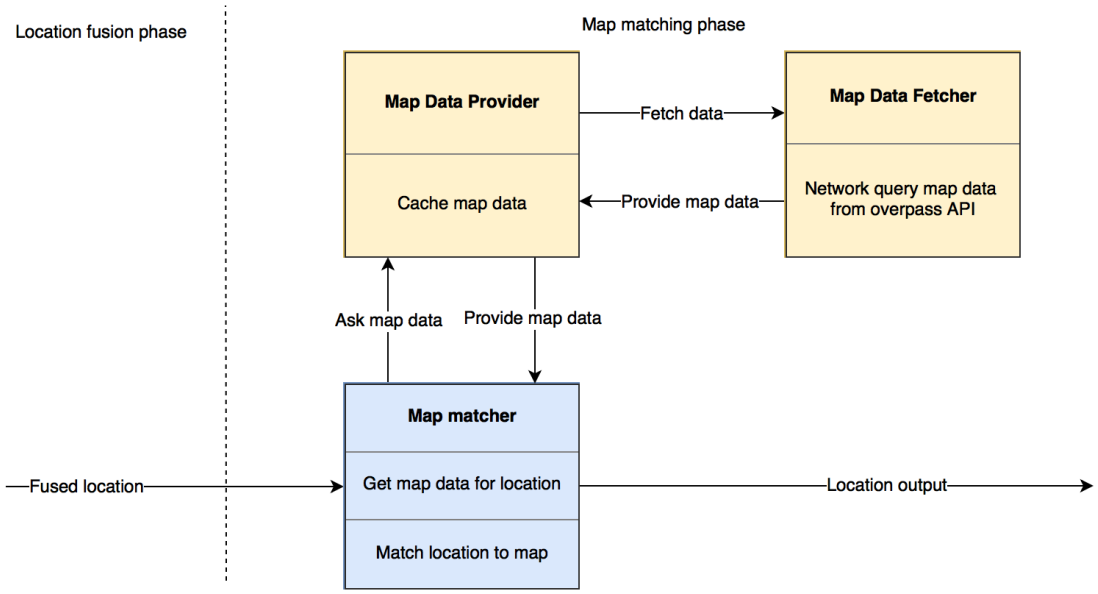


Figure 5. Archidecture of the map matching phase

Map Data Provider: An interface class with a task to provide map data to the map matcher module. For example, implementations can cache the already downloaded data for future use with various settings. An implementation is provided that caches map data that has already been retrieved from the Network once. Map data is cached as tiles. The tile size is configurable as is the maximum cached tiles count. The configured amount of map tiles are kept in memory simultaneously, and the latest retrieved map tile will be set to head of the queue. This configuration makes sure that tiles that were used last will be evicted from the queue first. Configuration is necessary to optimise the network download times and memory requirements of the mobile application.

Map Matcher: The main module for map matching. Receives the location data from the fused location phase and then processes it and outputs a map matched result. It first asks for a map tile (a rectangle section of a map) that falls within the location area to match the location with. A map tile includes the road network as lists of coordinate points. If there is no data right now, then the processing is skipped, and the location fused phase location is outputted. If the data is provided, the location is matched to the preexisting, and an enhanced location estimation is provided.

This is the final phase of location fusion and tracking algorithm. Next, the outputs can be displayed to the user.

3.3.4 Location output

There are two different outputs of the data. The first one is a visual output to the screen where the user can see the outputs in real-time. The visual output was primarily used for development purposes and to see how well the algorithm performed. Secondly, a text logging output was needed for further analysis of the data and to calculate some results. The user interface of the visual output of the algorithm is displayed in Figure 6. The Android UI composes of a map, control elements and data stream textual display.

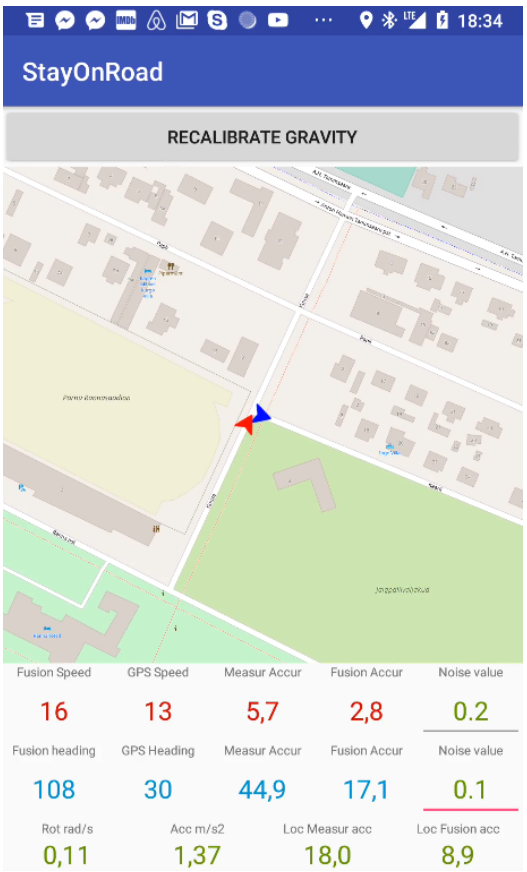


Figure 6. UI of the Android application

Map: The map used in the application is an android library OSMDroid [1]. OSMDroid uses OpenStreetMap data to create a map view for android mobile devices. It has various possibilities to add overlays to the map like Map markers, polylines and other simple geometric instances. In this application, two markers are put on the map: A red GPS output marker and the blue system output marker.

Control inputs: There are a total of 3 different control inputs that allow manipulating various algorithm parameters. Firstly the recalibrate gravity button, which is used to calibrate the gravity when launching the app and when the position of the device changes within the vehicle. The calibration of gravity is to ensure, that the gravity input can be removed from the accelerometer data stream. Secondly, there are two parameter text inputs. These are used to edit the process noise parameters of the speed and heading. Process noise is discussed in the next section.

Text data outputs: These are used mostly for debugging purposes. The first row of red outputs is used to output various speed parameters. For example, the algorithm speed estimation can be compared with the vehicle's speedometer, or see how well it responds to vehicle accelerations and decelerations. The second blue row of text outputs is dedicated to the heading estimation. Again, the algorithms heading estimate can be compared with the GPS outputs. The bottom row of green outputs are the filtered outputs of the gyroscope and accelerometer and the last two the GPS and location accuracy estimations.

This section discussed the overall architecture of the application. Next chapter describes the different modules of the algorithm and reasons their functionality.

4 Modules and Components

Based on our system architecture described in the previous section, the algorithms and methods integrated into our real-time system's modules are structured as in the next following subparagraphs.

4.1 Gravity

In order to get linear acceleration data, the gravity must be isolated from the data stream. Furthermore, gravity allows us to determine the rotational position of the phone with reference to Earth (Figure 7).

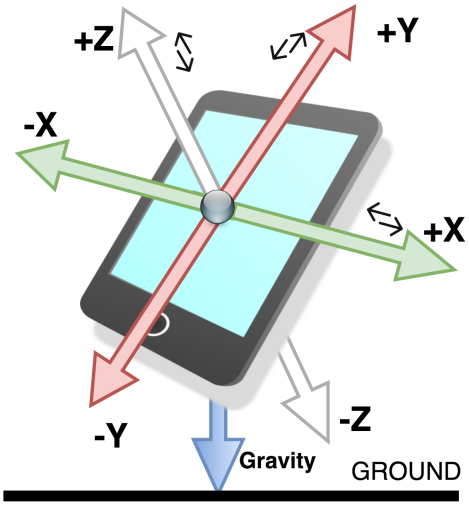


Figure 7. Gravity forces on Smartphone's accelerometer

Android provides a dedicated software sensor for accessing gravity data on the smartphone. However, this does not provide a satisfying degree of accuracy as the gravity is assumed to be constantly changing in the Android gravity sensor due to rotations of the smartphone. To get more accurate linear acceleration data, we have assumed that the phone has a fixed position inside the vehicle. The fixed nature of the smartphone inside the vehicle allows the user to recalibrate the gravity output while the vehicle is at a standstill and on an even field. Ideally, this needs to be done once after fixing the smartphone inside the vehicle.
Once the user starts the calibration sequence, a number of samples of accelerometer data are collected, and a moving average filter is applied with the priority given to the earlier data. First, we denote the gravity state vector as $\vec{g_i}$ and the measurement vector as $\vec{m_i}$, where the vectors represent the gravity strength values on x, y, z axes. Therefore, the

new estimated gravity state vector is defined as follows:

$$\vec{g_{i+1}} = \vec{g_i} + \frac{\vec{m_{i+1}}}{i+1} \quad \forall i \in \mathbb{N}; \tag{1}$$

Since there are no other acceleration inputs to the accelerometer other than gravity during the calibration phase, the data stream outputted by the accelerometer is an accurate gravity strength estimation on all x, y, z axes. Moreover, the gravity direction and strength on x, y, z axes can be used to alter other data.

4.2 Vehicle acceleration

GPS speed data is highly offset when there are big changes in speed [5]. Besides, the frequency of GPS in modern smartphones is usually $1hz$, which is too slow for an accurate real-time application. Therefore, to get the best estimation of speed as possible, there is a need for data fusion by extracting the acceleration from another sensor, such as the accelerometer (Figure 8).

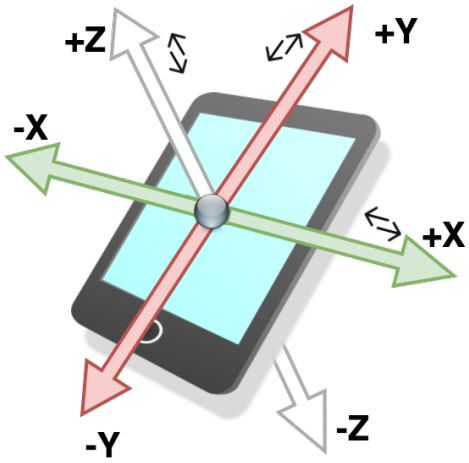


Figure 8. Smartphone's accelerometer measures the acceleration

Besides, to the phone having a fixed position in the vehicle, a requirement is made that the base of the phone is pointing directly towards the front of the vehicle (Figure 16), which is the usual positioning of the phone inside vehicles. This positioning requirement is needed because it would be complicated to separate the centripetal force from turns and the acceleration forces when the phone would be in an arbitrary position within the vehicle.

Obtaining acceleration data: Vehicle acceleration is obtained from the accelerometer by modifying it in the following way. Firstly, the gravity from the gravity module $\vec{g}$ is isolated from the accelerometer data vector $\vec{a}$ to acquire the linear accelerations $\vec{a_l}$, by:

$$\vec{a_l} = \vec{a} - \vec{g}; \tag{2}$$

The phone inside the car can be tilted and thus the acceleration forces would reflect on both the Z and Y axes, which would make the algorithm underestimate the acceleration forces if the unmodified data would be used from the Z axis. Hence, it is needed to reflect the vehicle acceleration and deceleration on the Z axis only. To accomplice this, the $\vec{a_l}$ is rotated by $\vec{g}$ with and angle β (Figure 9). The computation of the new value of

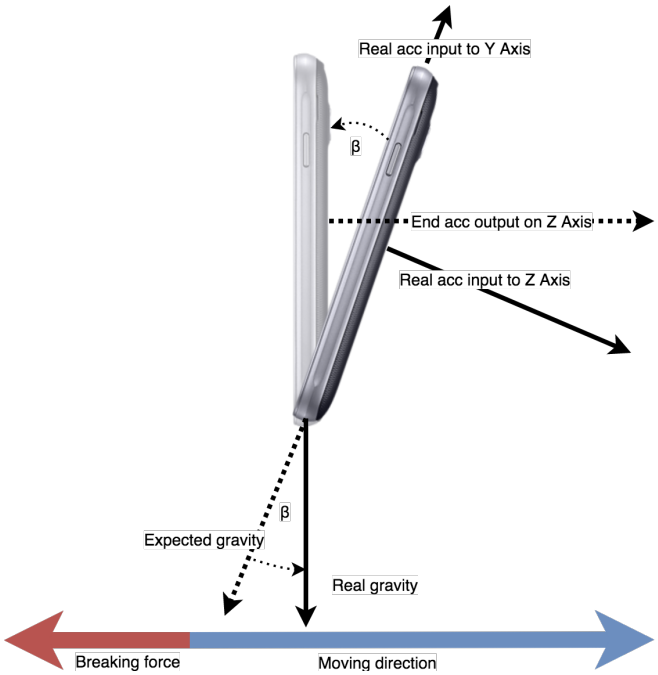


Figure 9. Smartphone's linear accelerometer data rotation

the acceleration $\vec{a_l}$ is done using the quaternion rotation formula [11]. Since we know that the force generating this rotation is the gravity $\vec{g} = g_x\vec{i} + g_y\vec{j} + g_z\vec{k}$, we will apply Euler's formula to compute the quaternion:

$$q = \exp\frac{\beta}{2}(g_x i + g_y j + g_z k) = \cos\frac{\beta}{2} + (g_x i + g_y j + g_z k)\sin\frac{\beta}{2} \tag{3}$$

Then, we evaluate the conjunction and compute the new value of the acceleration by applying the following formula:

$$\vec{a_l'} = q\vec{a_l}q^{-1} \tag{4}$$

After the modifications are applied, a low pass filter is performed to smooth out the data. A constant s is defined, which acts as a smoothing parameter.

$$\vec{a_{l_{i+1}}} = \vec{a_{l_i}} + \frac{\vec{m_{i+1}}}{s}; \tag{5}$$

4.3 Angular speed

Accurate tracking of vehicles turns is essential for localisation and tracking of the vehicle. Without keeping track of the moving direction of the vehicle, a drift occurs in location estimation every time the vehicle turns (Figure 10). GPS is too slow in most smartphones to respond to vehicle's headings changes and very unreliable at slow speeds under 3 m/s. Hence, a method is needed to observe vehicles angular movements. The solution

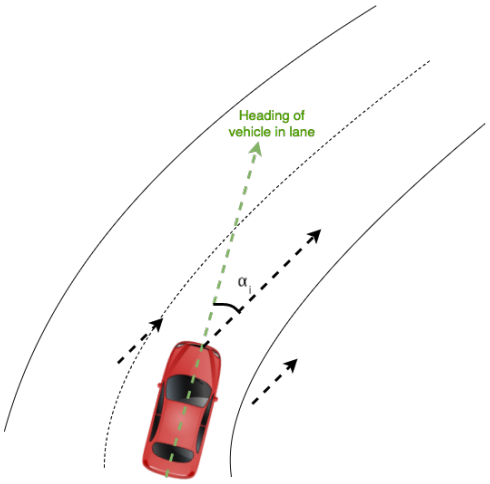


Figure 10. Heading drift

to this problem relies on the use of gyroscope sensor. The Gyroscope, which measures the rotations of the smartphone in 3D space (Figure 11), are included in most modern smartphones. Using the gyroscopes datastream, and by taking into account that our smartphone is in an immobile position, we can easily estimate with good accuracy the vehicle's turns.

Obtaining rotational data: As the phone can be tilted and positioned irregularly in the vehicle; thus the vehicles turns can be reflected on multiple axes. Isolating the vehicles turns into one data stream is essential for accurate turn estimations. As with the

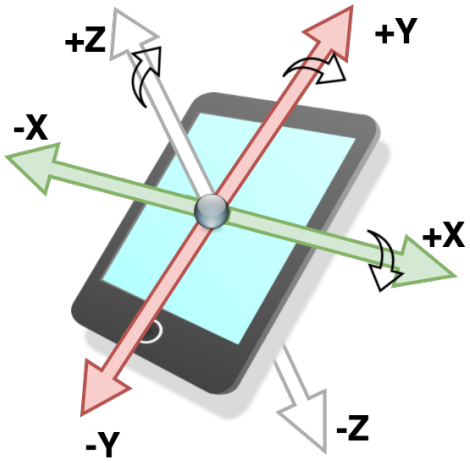


Figure 11. Smartphone's gyroscope measures rotation

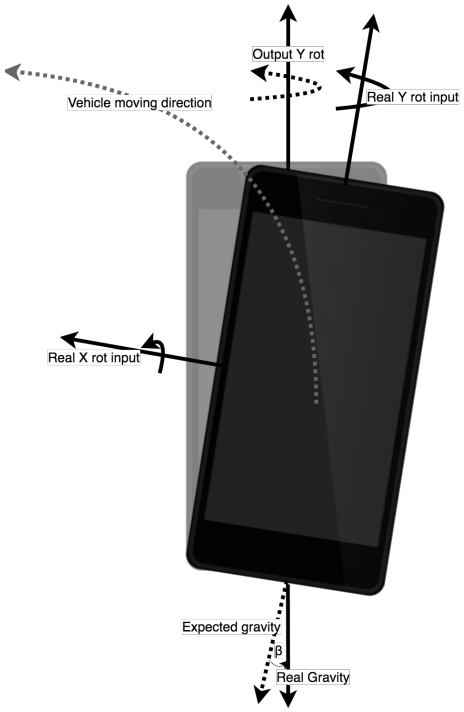


Figure 12. Smartphone's gyroscope measures rotation

acceleration module, the data is rotated using gravity in a similar way (Figure 12). As a consequence, the rotation isolates the vehicle turn inputs into Y axis. Now the data from the Y axis can be used to estimate the turns of the car. In the end, a low pass filter is applied similarly as in acceleration module to smooth out the data stream.

4.4 Speed fusion

Speed is a vital part of accurate estimation of location. With speed, it is possible to interpolate the position in real time when GPS updates are not frequent enough. Speed estimation is acquired by fusing the GPS speed from the GPS sensor and the vehicle acceleration from the acceleration module using Kalman filter. GPS speed updates act as measurement inputs to Kalman filter and the data from vehicle acceleration module acts as a control variable for the prediction phase of Kalman filter. Some definitions have to be made to apply Kalman filter to estimate speed.

Average acceleration definition: Firstly, we will start by defining the concept of average acceleration $\overline{a}_{\Delta t}$, which is a weighted average of acceleration measurements between times t_i and t_j, where time difference acts as a weight. Hence, our average acceleration is computed as follows:

$$\overline{a_{i,j}} = \frac{\sum_i^{j-1}(t_{i+1} - t_i) * a_i}{t_j - t_i} \tag{6}$$

Measurement error estimation: Moreover, Kalman filter requires a good error estimation of measurements. Most Android smartphones unfortunately do not offer speed error estimations form the GPS data stream. Therefore, a method was required to estimate the error in GPS speed updates. Since GPS speed updates are accurate when the vehicle is moving at a constant speed, but offset once some acceleration is applied, the proposed method to estimate the measurement error e_m is defined as follows:

$$e_m = (t_2 - t_1) * \overline{a_{t_1 t_2}}; \tag{7}$$

where t_1 is the last measurement time and t_2 is the current measurement time. The errors in speed measurements are equal to the average acceleration from the last measurement. For example when the last measurement was $30kph$, and the new measurement comes in 3 seconds later, with an average acceleration of $5kph/s^2$, then the error of the new estimation is $15kph$. An opposing example would be that if the last measurement was done at 30kph and the new measurement comes in 3 seconds later with an average acceleration of $0.5kph/s^2$, then the error would be estimated to be only $1.5kph$.

Process error definition: Kalman filter requires the definition of process error, which we introduce as e_p in speed estimation. It is defined as a constant:

$$e_p = 1m/s \tag{8}$$

This states that for every second the estimation can drift off to the real speed by $1m/s$.

Prediction and update phase: In order to constantly predict the state of speed v and the error e at time t_i, the Kalman filter prediction equations are used in the following manner:

$$v_{t_i} = v_{t_{i-1}} + (t_i - t_{i-1}) * \overline{a_{t_{i-1},t_i}} \tag{9}$$

$$e_{t_i} = e_{t_{i-1}} + (t_i - t_{i-1}) * e_p \tag{10}$$

Besides, new measurements v_m and measurement errors e_m are included into the speed state estimation v and error estimation e using the Kalman gain k as follows:

$$\begin{cases} k_t &= e_{t-1} * (e_{t-1} + e_m)^{-1} \\ v_t &= v_{t-1} + k_t * (v_m - v_{t-1}) \\ e_t &= (1 - k_t) * e_{t-1} \end{cases} \tag{11}$$

This ends our cycle for speed estimation and a speed estimation v can be output.

4.5 Heading's fusion

Heading estimation is used to avoid drifts while interpolating the location in real time with speed. It makes sure that the interpolation direction is constantly pointed in the right direction. The heading of the vehicle is estimated by fusing the GPS sensor outputs and the angular speed data outputs using the Kalman Filter. The GPS heading data acts as measurements, and the angular speed data acts as the control variable of the prediction phase of the Kalman filter. As with speed, most smartphones do not provide error estimations for the GPS bearing data.

Measurement error estimation: Since the heading accuracy and availability is highly dependent on speed in GPS measurements, the following quadratic function for error estimation e_m technique in GPS heading measurements is computed as follows:

$$e_m = \frac{90}{(speed_{m/s})^{1.5}} \tag{12}$$

The error measurement has a high value at slow speeds and quickly closes near 0 at higher speeds, which will help the Kalman filter to prefer measurement data in its predictions with higher speeds and disregard measurement data at low speeds.

Process error definition: The process error is defined as a constant, as with speed. It is assumed that for each second the estimation can drift up to $0.2deg/s$, which means that the Kalman filter will lean more measurement data when the measurements are more infrequent. Hence, we assign to the process error constant the value $0.2deg/s$

$$e_p = 0.2deg/s \tag{13}$$

Average rotation definition: Average rotation ($\overline{r_{i,j}}$) is defined as the average rotation in rad/s from time t_j to t_i

$$\overline{r_{i,j}} = \frac{\sum_i^{j-1}(t_{i+1} - t_i) * r_i}{t_j - t_i} \tag{14}$$

Prediction and update phase: The Prediction phase, where h is state of heading, e is the current error estimation, t_i is the current time, t_j is the last prediction time and $\overline{r_{ij}}$ is the average rotation from t_j to t_i:

$$h_i = h_{i-1} + (t_i - t_{i-1}) * \overline{r_{i-1,i}} \tag{15}$$

$$e_i = e_{i-1} + (t_i - t_{i-1}) * e_p \tag{16}$$

GPS heading measurements h_m and measurement errors e_m are input into the state and error estimations using the Kalman gain k:

$$\begin{cases} k_t &= e_{t-1} * (e_{t-1} + e_m)^{-1} \\ h_t &= h_{t-1} + k_t * (h_m - h_{t-1}) \\ e_t &= (1 - k_t) * e_{t-1} \end{cases} \tag{17}$$

After all of these processes the speed estimation is done for this cycle and an estimation of heading h is output.

4.6 Tracking and localisation fusion

The final step in the algorithm is to put all of the data together to achieve localisation and sensor fusion. Once again, Kalman Filter is used for fusing the data. GPS location lat, lng updates act as the measurements data acts as measurements and the already fused speed and heading act as control variables.

Converting speed and heading to latitude, longitude speed: As the speed and heading are output respectively as m/s and deg from the speed and heading modules then they must be converted to latitude and longitude speeds. The need for conversion to latitude and longitude speeds comes from the fact that the location inputs use earth coordinates. The speed and heading will be converted to v_{lat} and v_{lng} - given the speed v, heading h, current point lat_1, lon_1 and the earth's radius R. Firstly we calculate the lat_2 and lng_2 point after 1 second of movement given the previous parameters. The formula used for this purpose is as follows [26]:

$$\begin{cases} lat_2 &= \arcsin(\sin(lat_1) * \cos(d/R) + \cos(lat_1) * \sin(d/R) * \cos(h)); \\ lon_2 &= lon_1 + \arctan 2(\sin(h) * \sin(d/R) * \cos(lat_1), \cos(d/R) * \sin(lat_1) * \sin(lat_2)); \end{cases} \tag{18}$$

Finally, to get the latitude speed v_{lat} and longitude speed v_{lon} per second, we subtract from the future point the current point.

$$\begin{cases} v_{lat} & = lat_2 - lat_1 \\ v_{lon} & = lon_2 - lon_1 \end{cases} \tag{19}$$

Measurement error definition: The measurement error err is outputted on most Android devices. From the Android documentation [14] the accuracy is defined as a 68% confidence radius with respect to the real location. In other words, if a circle is drawn from the outputted location with a radius of the outputted error, there is a 68% chance that the real location falls within that circle. As the measurement error is one dimensional and the measurements are two dimensional then the measurement error E_m is defined as following:

$$E_m = \begin{bmatrix} err & 0 \\ 0 & err \end{bmatrix} \tag{20}$$

Process error definition: The process error e_p is defined as the possible drifted distance D away from the prediction, knowing the heading uncertainty from the heading module α, elapsed time from the last prediction Δt and the current estimated speed v and speed error e_v from the speed module (Figure 13). In other words, the error is described as, if the heading and speed error are both true, then we would drift away distance d from the real location. By using the law of cosines, the process error E_p can be calculated as

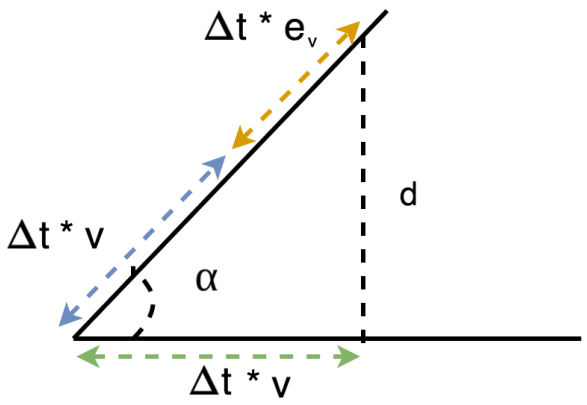


Figure 13. Process error estimation of the tracking and localisation fusion

follows:

$$d = \sqrt{(\Delta t * V)^2 + (\Delta t * v + \Delta t * v_e)^2 - 2(\Delta t * v) * (\Delta t * v + \Delta t * v_e) * \cos(\alpha_e)} \tag{21}$$

$$E_p = \begin{bmatrix} d & 0 \\ 0 & d \end{bmatrix} \tag{22}$$

Prediction and update phase: As stated before, working with latitude and longitude separately requires calculations in two dimensions. Therefore our state X and error estimation E will be defined as 2x2 matrices.

$$X = \begin{bmatrix} lat & 0 \\ 0 & lon \end{bmatrix}; \tag{23}$$

$$E = \begin{bmatrix} e_{lat} & 0 \\ 0 & e_{lon} \end{bmatrix}; \tag{24}$$

The prediction phase of location fusion is performed using the following formulas:

$$V = \begin{bmatrix} v_{lat} & 0 \\ 0 & v_{lon} \end{bmatrix}; \tag{25}$$

$$X_i = X_{i-1} + (t_i - t_{i-1}) * V \tag{26}$$

$$E_i = E_{i-1} + (t_i - t_{i-1}) * E_p \tag{27}$$

In addition, each observation can be included to our estimation by applying the measurement functions:

$$K = E * (E + E_m)^{-1} \tag{28}$$

$$X_i = X_{i-1} + K * (X_m - X_{i-1}) \tag{29}$$

$$E_i = (I - K) * E_{i-1} \tag{30}$$

After applying all of these equations the location can be outputted for this cycle. This was the last step in the Localisation and tracking phase. Next up is the Map matching module descriptions.

4.7 Map-matching

The reason for using map-matching in our system is meant to enhance the location accuracy visually to the user and possibly to reduce errors of the process. As maps can have errors as well, it cannot be stated that map-matching will reduce the error of the process as there could be errors in the mapping data as well. It might reduce errors

in the location estimation, although it improves human visuals and readability. Visual improvement of location is reached by matching locations to the adjacent road links by orthogonal projection [19]. After the map data has been queried from the Map Data Provider and a "MapTile" object has been received, then the map matching can begin.

Definitions: To define how the map-matching algorithm works, we have to define the road network model. First, let's annotate $V \in \mathbb{R}^2$ as nodes. Each node has two coordinates, latitude and longitude. $E \in V \times V$ as the edge segments connecting the nodes. As a result, we get a directed graph $G =< V, E >$ which defines our road network. We also assume that the road links are represented as straight segments, which is true for OSM data [9]. For each outputted location g_i from sensor fusion phase we perform in four stages:

- Extracting the road link candidates set S_i within a circle (g_i, r) with centre g_i and radius r that represents the area of 3 standard deviations of outputted location error,
- Calculate the difference of segment angle and heading angle and apply a likelihood function,
- Applying an orthogonal projection to the selected candidates' c_i from the road segment v_i and calculate likelihood with a function,
- Multiply the heading and distance likelihood and emit the maximum likelihood projection.

Segment extraction: Segment extraction is performed by applying the following equation:

$$S_{e_i,r} = \{\, e \mid e \in E \wedge\ proj(g_i, e) \leq r \,\}\,; \tag{31}$$

where, $proj(,)$ is the minimum distance between a location and a segment e, aka *orthogonal projection* distance. For each extracted road segments $e_i \in L$, two evaluations are applied. Firstly the orthogonal projection is calculated between the segments and the point g_i. Secondly, the segment angle and the heading angle are compared, and a likelihood is calculated. The latter helps to avoid matching one-way segments which might run in the opposite direction than the current heading. Also, it helps to match the right segment near intersections, where the orthogonal projection alone might pick the wrong segment.

Distance likelihood set: Distance set $D_{g_i,r}$ is computed using an *orthogonal projection* on all the edges in set $S_{e_i,r}$ (Figure 14) and Distance likelihood L_{d_i} set has the likelihood function f_d applied to the distance set:

$$D_{g_i,r} = \{\ proj(g_i, e_j) \mid \forall e_j \in D_{g_i,r}\}\ ; \tag{32}$$

$$f_d(x, r) = argmin(1,\ argmax(0,\ (1 - \frac{x}{3 * r})\ ; \tag{33}$$

$$L_{d_i,r} = \{\ f_d(d, r) \mid \forall d \in D_{g_i,r}\}\ ; \tag{34}$$

$$\tag{35}$$

Heading likelihood set: To calculate the heading likelihood set, we first must acquire the segment heading h_e. The bearing between segment starting point (lat_1, lng_1) and ending point (lat_2, lng_2) aka, two geographical locations is calculated as follows.

$$lng_d = lng_2 - lng_1; \tag{36}$$

$$h_e = atan2(\sin(lng_d) * cos(lat_2), cos(lat_1) * sin(lat_2) - sin(lat_2) * cos(lat_2) * cos(lng_d)); \tag{37}$$

Now, the segment heading calculation can be applied to all the edges in $S_{e_i,r}$ and the Likelihood function f_h can be used on the segments in the Heading set H.

$$H_{g_i,r} = \{\ h_{e_j} \mid \forall e_j \in S_{g_i,r}\}\ ; \tag{38}$$

$$f_h(x) = argmin(1,\ argmax(0,\ (1 - \frac{x}{90}) \tag{39}$$

$$L_{h_i,r} = \{\ f_h(h) \mid \forall h \in H_{g_i,r}\} \tag{40}$$

Selecting segment: Afterwards the likelihood sets can be combined by multiplying all the corresponding edge likelihoods (element wise set multiplication) and the maximum likelihood segment will be selected. Finally, the point that is projected onto the selected segment will be output.

$$proj(g_i, v_j) = argmax\ \ x_i \quad \forall x_i \in L_{h_i,r} \bullet L_{d_i,r}; \tag{41}$$

This ends the Module section of this thesis. In the following section the Results of testing the system are displayed and discussed.

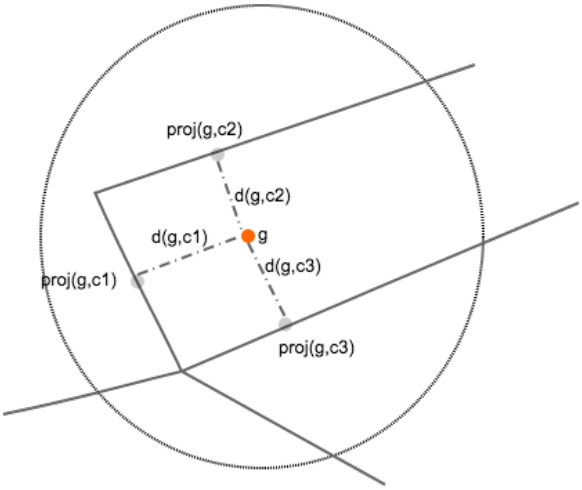


Figure 14. Map-matching Algorithm: Projection of GPS point g and distance to road segments candidates c_i within defined GPS error circle

5 Results and Discussion

This section discusses the evaluation of the project and displays the results. The following aspects of the algorithm are presented, firstly the response times in the real-time point of view. Secondly, the accuracy of the algorithm is put to the test in various conditions.

5.1 Experiment setup

For the testing, the following equipment was used

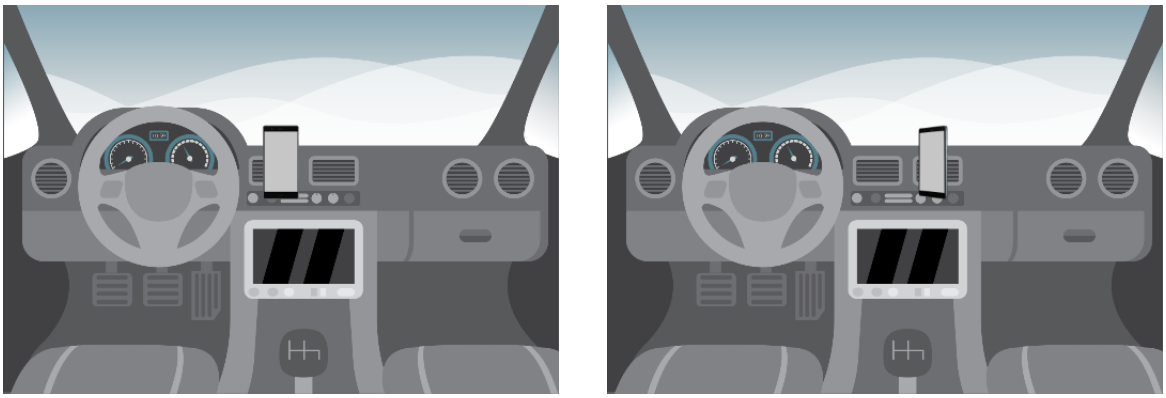
- Nexus 5X Smartphone with Android 8.0.0 operating system
- Dedicated Brodit phone cradle for Nexus 5X (Figure 15), to eliminate any factors of bad fit phone cradle.
- Sporty car, to test out the algorithm under harsher breaking, accelerations and turns.

The Brodit phone cradle was mounted into the vehicle as required: The base of the phone or in other words the back cover was set straight into the moving direction, to eliminate inputs from turns onto the accelerometer Z axis. Figure 16 displays a correct and incorrect mounting of the phone inside the vehicle.

The route used for testing and driving was a 1,98km long city course along the streets of Võru, Estonia. The path taken is illustrated in Figure 17. The route has quite an extensive array of common road characteristics, like long straights, sharp turns,

long smooth turns, short straights with short bursts of acceleration and deceleration, a roundabout and almost a U-turn, which are perfect to test out the performance of the algorithm.

Figure 15. Nexus 5X smartphone and Brodit phone cradle



(a) Case of a correct placement inside the vehicle. The base of the phone is in the same direction as the vehicle moving direction

(b) Case of an incorrect placement inside the vehicle. The base of the phone is at an angle of the vehicle moving direction

Figure 16. The correct and incorrect placement of smartphone inside the vehicle

5.2 Evaluation strategy

The real-time system and its estimation techniques were put to the test using real-world conditions. The route was driven by a road legal vehicle during normal business hours, using only publicly available devices like the Nexus 5X smartphone and the Brodit smartphone cradle. To test the performance of our system with different sparseness

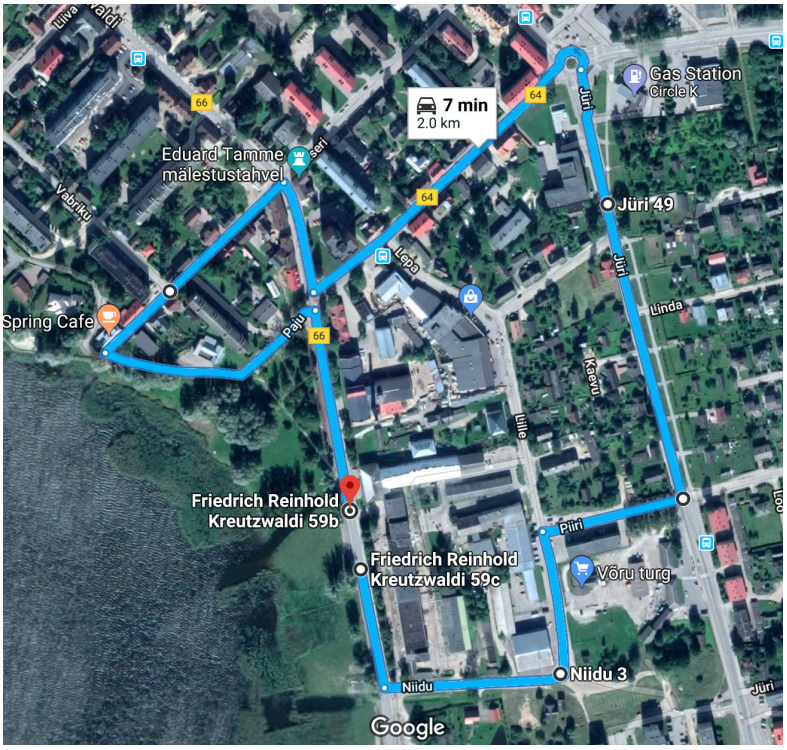
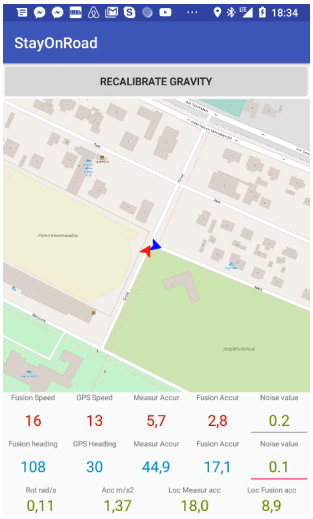


Figure 17. A Google Maps satellite image of the common route driven during the test period

of measurements in time, the route was driven for four times with four different GPS update rate inputs ($0.4Hz$, $0.2Hz$, $0.1zHz$, $0.033Hz$). The different frequency rates were chosen to see how the technique was behaving. In order to evaluate our system besides the visual evaluation, a separate GPS data stream of 1HZ was logged separately. This allows the comparison between the localisation and tracking predictions and the GPS 1HZ datastream.

5.3 Experimental Results

The results of our testing of the four different GPS frequencies ($0.4Hz$, $0.2Hz$, $0.1zHz$, $0.033Hz$) are illustrated in Figure 19. The map shows the path traces of what the system estimated in real-time. The red line represents the location estimations by multi-sensor fusion. The blue line shows the location after the map-matching process. And green markers with arrows show the GPS measurements used as inputs to our system. The outcome shows that even when there is no GPS input for up to 30 seconds, our real-time system is still capable of providing a location estimation with acceptable accuracy.



(a) Case of a vehicle having the intention to turn to the right

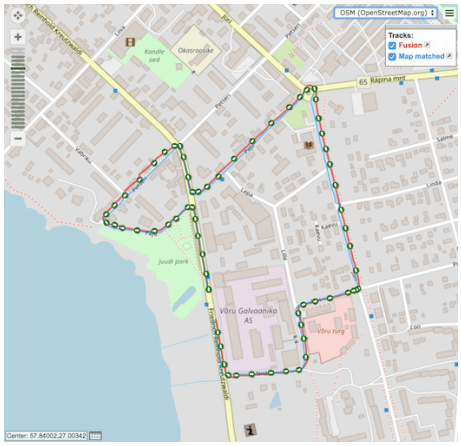
(b) Case of a vehicle moving in a straight road segment

Figure 18. Overview of the real-time system (at t and t+1) - red arrow is the GPS measurement and blue arrow is the prediction (Tracking)

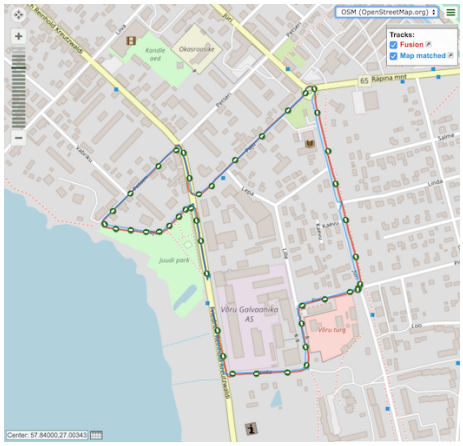
Figure 20 shows the distance difference in meters between our system output and the latest logged GPS 1HZ output. The graph presents three different metrics used for evaluation:

- The error measurement shows the 1HZ GPS location errors outputted by the Android system.
- The fusion difference shows the difference in meters between the predicted location and the GPS outputs.
- Map matched difference is the distance between each outputted GPS location and the predicted map-matched location.

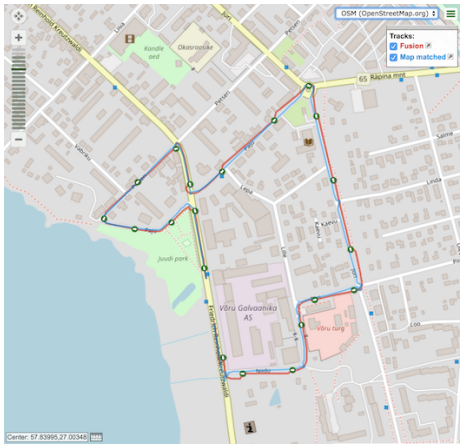
We can notice that the moment the frequency of GPS inputs gets smaller, the accuracy of our estimation for localisation and tracking becomes slightly lower. The drop in accuracy is expected because drift occurs between the measurement inputs and only new inputs can correct that error. It must be noted, that most of the drift occurs due to the fact of bad speed estimation. While the estimation of heading is quite accurate, the speed estimation performs more poorly. The poor speed estimation is partly because inclinations of the road affect the acceleration estimation due to changing gravity and partly because the Accelerometer data stream is noisy. The results also show that the map-matching in average gives sometimes good results and other not, since the measurement has errors and also the used map-matching is very simplistic, and it can be made better



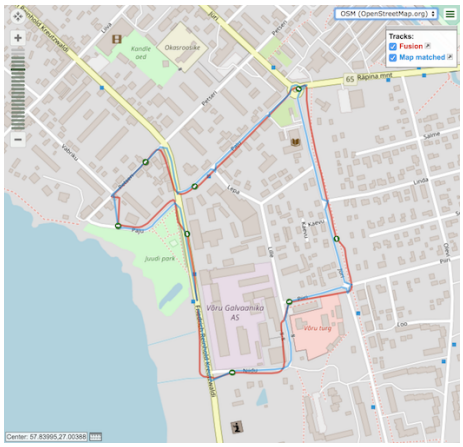
(a) Testing with 0.4hz frequency GPS



(b) Testing with 0.2hz frequency GPS



(c) Testing with 0.1hz frequency GPS

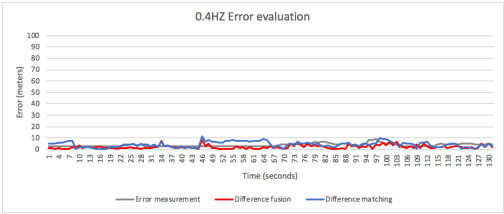


(d) Testing with 0.033hz frequency GPS

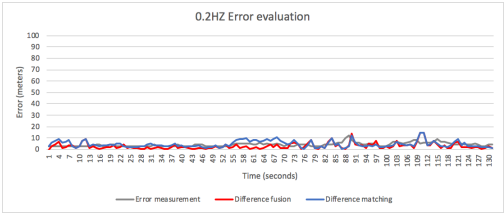
Figure 19. Overview of the system's results on a map - After fusion process and also after Map-matching process

(Table 2). Map-matching is also affected by the poor speed estimation because while it can keep the estimation on the right road, the fact how far on the road the vehicle has travelled is directly connected with the speed estimation.

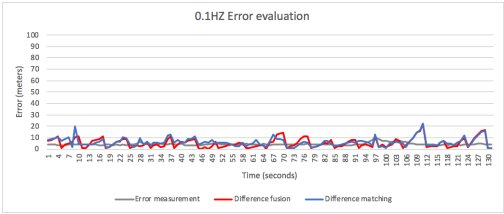
The computational time of our system is represented in table 3. The multi-sensor fusion needs approximately 2ms to finish the execution, and the map-matching process requires about 1ms, which means the whole system needs around 3ms to show the final results on the map. In general, the system has a very acceptable performance as a real-time system for localisation and tracking.



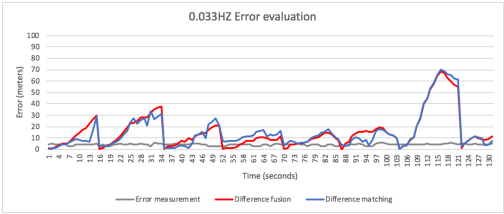
(a) Evaluation of 0.4hz GPS inputs



(b) Evaluation of 0.2hz GPS inputs



(c) Evaluation of 0.1hz GPS inputs



(d) Evaluation of 0.033hz GPS inputs

Figure 20. Overview of the system's evaluation results - Difference between 1HZ GPS data stream and system's outputs in meters

Frequency	0.4HZ	0.2HZ	0.1HZ	0.033HZ
Average measurement error	3.76m	4.23m	4.2m	4m
Average fusion difference	2.09m	2.92m	5.15m	14.18m
Average map matching difference	4.21m	4.43m	6.03m	13,96m

Table 2. Average evaluation results for each GPS measurement frequency

Task	Mutli-Sensor fusion	Map Matching process	Total
Measurement cycles	2269	2228	4497
Average time	2.02ms	1.14ms	3.16ms

Table 3. Time response of the system and the measurement cycles

6 Conclusion

In this thesis, a software is presented that can take advantage of the many sensors inside a smartphone and localise and track vehicles. The application is installed on a device with an Android OS, and the sensors that are exploited for localisation and tracking are the GPS, accelerometer and gyroscope. They all give out data at different rates and must be fused together to achieve the best localisation and tracking system. Kalman filter is used in the proposed system to fuse the data together. GPS data stream is used as a measurement and accelerometer and gyroscope data stream is used as control variables for speed changes and turns respectively. Finally, to achieve better human readability,

the outputted location is map-matched to the displayed map. As the map is composed of straight segments, then the most likely segment must be chosen, where the user is. The segment is chosen by calculating the location and segment distance and heading likelihood. When the likelihoods are calculated for both distance and heading, then the respective numbers are multiplied, and the segment with the highest likelihood is chosen. The last step in map matching is to project the point to the segment and output that projection.

Results: The software can respond to sudden speed changes satisfactorily and exceptionally well to vehicle turns. Even if the GPS signal is lost for 30 seconds or more, then the location estimation error remains on average below 30m accuracy. It must be noted that the most significant error accumulation comes from the aspect of incorrect speed estimation. Also, the processing time is for each cycle of location estimation is on average around 3ms, which means that the application can process more than 300 estimates per second, which is well above expected. In conclusion, the system achieves all the goals set at the beginning of development and even exceed expectations.

Future work: In the future, the map matching algorithm could be fine-tuned and developed further with more sophisticated methods to achieve even better map matching results. Secondly, the speed estimation methods are disturbed by inclinations, so that must be taken into account with some techniques. Perhaps an OBDII interface with the car could be exploited to get the speed input from the vehicle. The use of OBDII speed output would significantly increase the accuracy of speed estimation.

The system gives out promising results and accurate results with even lousy GPS reception. With some work, it could be easily used in production environments.

References

[1] Osmdroid github webpage. https://github.com/osmdroid/osmdroid. [Online; accessed apr-2018].

[2] Overpass turbo api. https://overpass-turbo.eu/. [Online; accessed may-2018].

[3] C. Beder and M. Klepal. Fingerprinting based localisation revisited: A rigorous approach for comparing rssi measurements coping with missed access points and differing antenna attenuations. In *Proc. Int. Conf. Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–7, November 2012.

[4] Pierre Carbonnelle. Pypl popularity of programming language. http://pypl.github.io/PYPL.html. [Online; accessed feb-2018].

[5] Rob Duffield, Machar Reid, John Baker, and Wayne Spratford. Accuracy and reliability of gps devices for measurement of movement patterns in confined spaces for court-based sports. *Journal of Science and Medicine in Sport*, 13(5):523–525, 2010.

[6] J. E. Everett. The exponentially weighted moving average applied to the control and monitoring of varying sample sizes. In *Computational Methods and Experimental Measurements XV*. WIT Press, may 2011.

[7] Xiaocong Fan. *Real-Time Embedded Systems: Design Principles and Engineering Practices*. Newnes, 2015.

[8] OpenStreetMap Foundation. Openstreetmap about page. https://www.openstreetmap.org/about. [Online; accessed mar-2018].

[9] Jean-François Girres and Guillaume Touya. Quality assessment of the french openstreetmap dataset. *Transactions in GIS*, 14(4):435–459, 2010.

[10] Shan He, Sizhao Li, Yan Chen, and Donghui Guo. Uncertainty analysis of race conditions in real-time systems. In *Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on*, pages 227–232. IEEE, 2015.

[11] Noel H Hughes. Quaternion to euler angle conversion for arbitrary rotation sequence using geometric methods. *Accessed online at noelhughes. net/uploads/quat*, 2, 2008.

[12] Hyeongsoon Im, Bonghee Hong, Seungwoo Jeon, and Jaegi Hong. Bigdata analytics on cctv images for collecting traffic information. In *Proc. Int. Conf. Big Data and Smart Computing (BigComp)*, pages 525–528, January 2016.

[13] K. Jo, K. Chu, and M. Sunwoo. Interacting multiple model filter-based sensor fusion of GPS with in-vehicle sensors for real-time vehicle positioning. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):329–343, March 2012.

[14] Google LLC. Android location class documentation. `https://developer.android.com/reference/android/location/Location.html#getAccuracy()`. [Online; accessed apr-2018].

[15] Google LLC. Sensors overview. `https://developer.android.com/guide/topics/sensors/sensors_overview.html`. [Online; accessed mar-2018].

[16] D. Lymberopoulos and J. Liu. The microsoft indoor localization competition: Experiences and lessons learned. *IEEE Signal Processing Magazine*, 34(5):125–140, September 2017.

[17] A. Makki, A. Siddig, M. Saad, J. R. Cavallaro, and C. J. Bleakley. Indoor localization using 802.11 Time differences of arrival. *IEEE Transactions on Instrumentation and Measurement*, 65(3):614–623, March 2016.

[18] R. Momose, T. Nitta, M. Yanagisawa, and N. Togawa. An accurate indoor positioning algorithm using particle filter based on the proximity of bluetooth beacons. In *Proc. IEEE 6th Global Conf. Consumer Electronics (GCCE)*, pages 1–5, October 2017.

[19] Washington Y Ochieng, Mohammed Quddus, and Robert B Noland. Map-matching in complex urban road networks. *Revista Brasileira de Cartografia*, 2(55), 2003.

[20] C. C. Peng, Y. T. Wang, and C. L. Chen. LIDAR based scan matching for indoor localization. In *Proc. IEEE/SICE Int. Symp. System Integration (SII)*, pages 139–144, December 2017.

[21] Thanh Nam Pham, Ming-Fong Tsai, Duc Binh Nguyen, Chyi-Ren Dow, and Der-Jiunn Deng. A cloud-based smart-parking system based on internet-of-things technologies. *IEEE Access*, 3:1581–1591, 2015.

[22] I. Skog, P. Handel, J. O. Nilsson, and J. Rantakokko. Zero-velocity detection —an algorithm evaluation. *IEEE Transactions on Biomedical Engineering*, 57(11):2657–2666, November 2010.

[23] TIOBE software BV. Tiobe index for april 2018. `https://www.tiobe.com/tiobe-index/`. [Online; accessed feb-2018].

[24] Shun Taguchi, Satoshi Koide, and Takayoshi Yoshimura. Online map matching with route prediction. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–10, 2018.

[25] Stephen Teang Soo Thong, Chua Tien Han, and Tharek Abdul Rahman. Intelligent fleet management system with concurrent gps & gsm real-time positioning technology. In *Telecommunications, 2007. ITST'07. 7th International Conference on ITS*, pages 1–6. IEEE, 2007.

[26] Chris Veness. Calculate distance, bearing and more between latitude/longitude points. `https://www.movable-type.co.uk/scripts/latlong.html`, January 2015. [Online; accessed feb-2018].

[27] H. Xiong, J. Tang, H. Xu, W. Zhang, and Z. Du. A robust single GPS navigation and positioning algorithm based on strong tracking filtering. *IEEE Sensors Journal*, 18(1):290–298, January 2018.

[28] C. Yu, X. Li, L. Ju, Y. Zhang, J. Qin, L. Dou, and J. Liu. Time mode based next position prediction system. In *Proc. IEEE 19th Int. Conf. High Performance Computing and Communications; IEEE 15th Int. Conf. Smart City; IEEE 3rd Int. Conf. Data Science and Systems (HPCC/SmartCity/DSS)*, pages 586–593, December 2017.

[29] J. M. Ibarra Zannatha, R. Cisneros Limon, A. D. Gomez Sanchez, E. Hernandez Castillo, L. E. Figueroa Medina, and F. J. K. Lara Leyva. Monocular visual self-localization for humanoid soccer robots. In *Proc. st Int CONIELECOMP 2011 Conf. Electrical Communications and Computers*, pages 100–107, February 2011.

[30] H. Zou, Z. Chen, H. Jiang, L. Xie, and C. Spanos. Accurate indoor localization and tracking using mobile phone inertial sensors, WiFi and ibeacon. In *Proc. IEEE Int. Symp. Inertial Sensors and Systems (INERTIAL)*, pages 1–4, March 2017.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Siim Plangi**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   Real-time Localisation and Tracking System for Navigation Based Mobile Multi-sensor Fusion

   supervised by Dr Amnir Hadachi and Mr Artjom Lind

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 15.05.2018