UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Helena Sokk

Vision-based localization on city scale using Open Street Map

Master's Thesis (30 ECTS)

Supervisor(s): Naveed Muhammad, PhD
Dmytro Zabolotnii, MSc

Tartu 2025

Vision-based localization on city scale using Open Street Map

Abstract:
Autonomous vehicles, like other robots, need to localize themselves in order to navigate. While global navigation satellite systems (GNSS) such as GPS can provide such vehicles with localization information, the GNSS information might not always be available. Since localization is one of the crucial components in self-driving vehicles, it is important to develop robust techniques to accomplish it. One such localization technique for vehicles to localize is using particle filters, given a map of the environment.

The goal of this thesis was to implement a robust localization framework that integrates the particle filter with vision-based street sign detection and Open Street Map, without any reliance on GNSS. The proposed framework was able to localize the vehicle within a radius of 10 meters of its ground truth location, showing promising results.

The implemented framework provides a good starting point for any future improvements and experiments in the problem of GNSS-free localization in autonomous vehicles.

Keywords: self-driving vehicles, localization, particle filter

CERCS: T120 Systems engineering, computer technology, T125 Automation, robotics, control engineering

Nägemispõhine linnasisene asukoha määramine, kasutades Open Street Map'i

Lühikokkuvõte:
Isejuhtivad sõidukid, nagu ka teised robotid, peavad navigeerimiseks enda asukoha kindlaks määrama. Kuigi globaalsed satelliitnavigatsioonisüsteemid (GNSS) nagu GPS võivad pakkuda sõidukitele nende asukoha kohta teavet, siis GNSS-i poolt pakutav informatsioon ei pruugi alati olla kättesaadav. Kuna asukoha määramine on üks isejuhtivate sõidukite kriitilisemaid komponente, siis on tähtis töökindlate meetodite välja töötamine antud valdkonnas. Üks selline meetod sõidukite asukoha määramiseks on osakeste filtri kasutamine, kui on olemas kaarditeave ümbritseva keskkonna kohta.

Käesoleva lõputöö eesmärk oli rakendada kindlat positsioneerimisraamistikku, mis ühendab endas osakeste filtri, nägemispõhise tänavasiltide tuvastamise ja Open Street Map'i, ilma GNSS-i kasutamata. Kavandatud raamistik suutis sõiduki asukoha tuvastada 10 meetri raadiuses selle tegelikust asukohast, näidates paljutõotavaid tulemusi.

Rakendatud raamistik on heaks lähtepunktiks tulevikutäiustuste ja -katsete tegemiseks seoses isejuhtivate sõidukite GNSS-vaba positsioneerimisega.

Võtmesõnad: isejuhtivad sõidukid, asukoha määramine, osakeste filter

CERCS: T120 Süsteemitehnoloogia, arvutitehnoloogia, T125 Automatiseerimine, robootika, juhtimistehnika

Contents

1 Introduction

1.1 Context

Localization, or pinpointing location inside a map, is one of the fundamental problems of robotics, enabling mobile robots to navigate, map their surroundings and carry out various tasks autonomously [Ala23]. Being able to localize precisely is essential for the safety and effective operation of autonomous vehicles [KM23].

Early approaches of localization involved the use of global satellite navigation systems (GNSS). Despite the widespread adoption of such systems like global positioning system (GPS) they suffer from significant limitations in dense urban environments due to signal occlusion, multipath effects, and inaccuracies caused by building reflections [Kap96]. To overcome these limitations, the combination with other sensing methods was used - most commonly GNSS with dead reckoning. One such example was commercial Siemens car navigation system which implemented map matching with the information from GNSS, odometers, and gyroscopes. [WFMB19]

Other approaches involved the use of high-precision sensors, such as light detection and ranging (LiDAR), which provided detailed environmental data necessary for localization. In 2009, Waymo (formerly Google Car)[1] began developing autonomous vehicles, incorporating LiDAR as a primary sensor for localization and mapping which marked a significant step in the evolution of localization technology for autonomous vehicles. Waymo's approach involved integrating multiple LiDAR sensors to ensure a comprehensive view of the vehicle's surroundings, enhancing the accuracy of localization. [WFMB19]

Tesla[2], on the other hand, took a different approach relying on a combination of radar, sonar, IMU, and cameras for localization, aiming to reduce costs and still achieve reliable localization. Since 2024, Tesla began transitioning into their camera-based system, known as Tesla Vision, which relies solely on cameras for object detection and navigation [Tes25]. The development of HD (high definition) maps which provide static information about the road environment also played a crucial role in improving localization accuracy [WFMB19].

Recent advances in computer vision and the growth of geospatial data have opened new avenues for addressing satellite navigation challenges. Among these, vision-based localization, which uses the rich visual context of urban environments, has emerged as a promising alternative or complement to GNSS. By matching visual input from cameras to pre-existing georeferenced maps, it is possible to achieve more accurate and reliable localization, even in GNSS-degraded environments. [Afi14]

Open Street Map (OSM) [Wik25], a widely used open-source mapping platform, provides a vast amount of geospatial data that can be harnessed for vision-based localiza-

[1] Waymo, https://waymo.com/

[2] Tesla, https://www.tesla.com/

tion. Its detailed and freely available map data, which includes roads, building outlines, and landmarks, offers a versatile framework for integrating visual cues with geographic information.

1.2 Problem statement

Previous research has relied heavily on an existing or inaccurate GNSS position that cannot be taken for granted in conditions where a GNSS position is not available (for example, in case of a GNSS-sensor failure in an autonomous vehicle). This gap in literature creates an opportunity to delve into possible solutions for localizing an autonomous vehicle without using the GNSS position. Therefore, this thesis explores the integration of vision-based techniques with Open Street Map to achieve robust localization in urban conditions where GNSS is unavailable.

1.3 Contributions

The key contributions of this work are threefold: first, the development of a pipeline for matching data gathered from street view images to OSM-based representations of urban environments; second, the integration of a particle filter for probabilistic localization, enabling robust handling of uncertainties and dynamic scenarios; and third, the evaluation of the system's performance across diverse urban environments and a critical analysis of the scalability and limitations of using OSM for vision-based localization at city scale.

1.4 Structure of the manuscript

This thesis is organized as follows: Section 2 gives an overview of autonomous vehicles, machine learning algorithms, and background information about particle filters and Open Street Map. Section 3 explores related work on localization techniques in urban environments, discussing their strengths and weaknesses. Section 4 presents methodology part that includes assumptions, the setup of the framework, and the choice of evaluation metrics and tools. Section 5 presents the experimental results, evaluation, and discussion in a form of analysis. Finally, Section 6 concludes the thesis and discusses possible future work on this topic.

NOTE: No generative AI tool was used to generate any part of text in this manuscript (besides any AI that the default Overleaf spell and grammar check might be using).

2 Background

This chapter explains what are autonomous vehicles and how do they localize, what computer-vision algorithms are currently used for localization task, what are particle filters and what are the main challenges and problems. Finally, an overview is given of existing solutions to localization combining computer vision, particle filters and Open Street Map.

2.1 Autonomous vehicles

Autonomous vehicles, or often known as self-driving cars, represent a transformative innovation in the field of transportation. These vehicles are equipped with advanced sensors, computing power, and algorithms to navigate and operate without human intervention. They promise to revolutionize mobility by enhancing safety, improving traffic efficiency, and reducing the environmental footprint of transportation [YLCT19]. According to SAE, there are six levels of driving automation, ranging from no driving automation (Level 0) to full driving automation (Level 5) (see Figure 1).

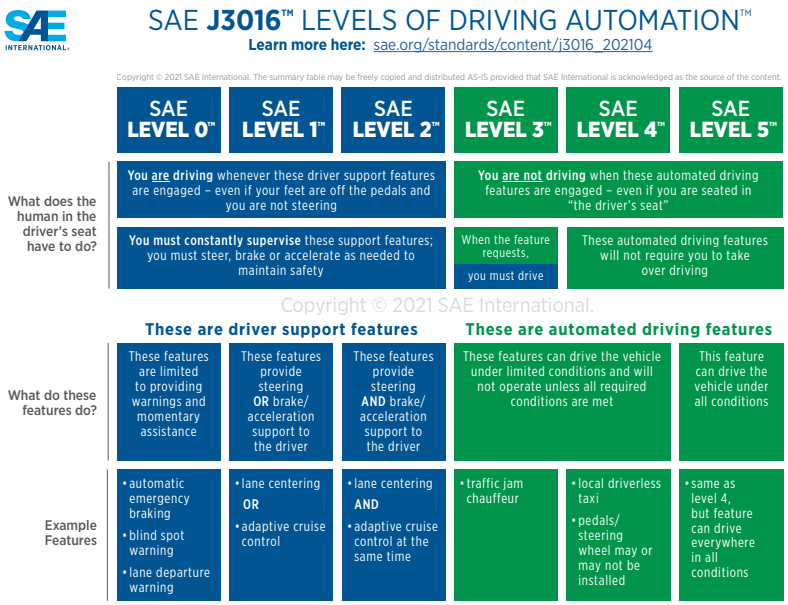


Figure 1. SAE J3016 Visual Chart showing six levels of driving automation, from [Int25]

Autonomous systems are rapidly evolving, with several companies leading the charge in deploying self-driving technology. Some examples of currently deployed autonomous

vehicles include robotaxis provided by such companies like Waymo, Zoox[3] and Lyft[4]. These robotaxis are becoming more prevalent, with Waymo expanding its testing to new cities. Another important field where automation can create better opportunities for reducing workload, time, and increase safety, is goods delivery. For example, companies like Aurora[5] have fully dedicated themselves into heavy truck automation. In addition, Uber has partnered with Nvidia to accelerate AI advancements in autonomous driving. These developments highlight the growing role of automation in transportation and logistics [Kor24, UT25].

2.1.1 Localization of autonomous vehicles

Several sensors and algorithms are used for localization in autonomous vehicles, including satellite navigation systems, vehicle motion sensors, range sensors, and vision sensors. These sensors provide data that is then processed using various algorithms such as Kalman filters, particle filters, and graph-based approaches [WFMB19].

A formal definition of localization reviewed by [KM23] states that localization is a critical function for autonomous vehicles, enabling them to accurately determine their position within their environment. The level of precision of the localization depends completely on the objective of the vehicle [WFMB19]. Accurate vehicle self-localization is significant for autonomous driving, especially in urban areas where Global Navigation Satellite System (GNSS) signals can be unreliable due to buildings partially blocking the sky (see Figure 2).

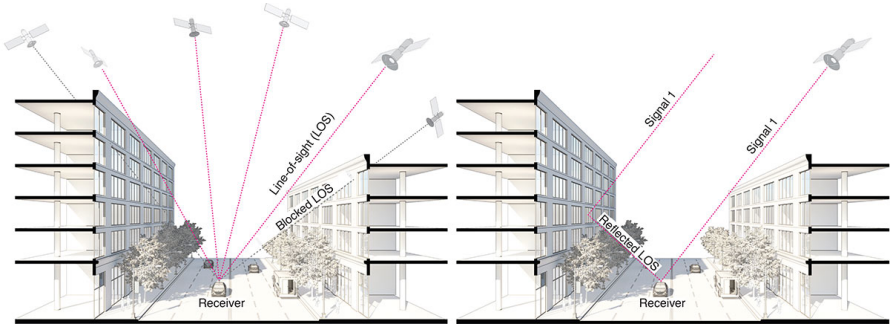


Figure 2. Three types of potential GNSS signal reception: direct LOS signals and blocked LOS signals (left) and reflected LOS signals (right), from [NSHP21]

Localization can be categorized into three separate components: Road-level lo-

[3]Zoox, https://zoox.com/
[4]Lyft autonomous rides, https://www.lyft.com/autonomous
[5]Aurora, https://aurora.tech/

calization, Ego-Lane-level localization, and Lane-level localization [LKA+22]. Each component gives different level of precision. For example, for lane-keeping it is not sufficient to use road-level localization since its accuracy is within meters. Localization techniques often involve matching what the vehicle observes to a priori known maps, either through visual odometry (VO) or Simultaneous Localization and Mapping (SLAM) methods [LKA+22].

Since the emergence of GPS in 1990s, the importance of map-matching has increased significantly. It is particularly visible in online map-matching, which is best used for real-time applications (see Figure 3). Map-matching is a crucial component in dealing with inaccuracies in vehicle positioning, which helps to identify the physical location of the vehicle.

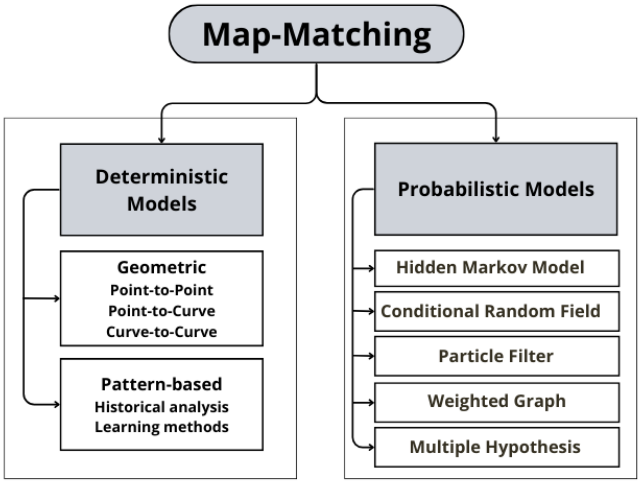


Figure 3. Map-Matching classification, splitting the techniques into two main parts that are the deterministic and probabilistic approaches. Adopted from [LKA+22]

Visual odometry (VO) is a relative localization technique that estimates the vehicle's movement by analyzing sequences of images. It can be categorized into appearance-based and feature-based approaches. SLAM methods, on the other hand, involve creating a map of the environment while simultaneously keeping track of the vehicle's location within that map. However, SLAM can suffer from error accumulation, making it challenging for high-speed environments like highways.

To improve localization accuracy and reduce storage requirements, some approaches use abstract maps that represent the urban environment with multilayer 2D vectors and 3D planar surfaces, significantly shrinking the map size while preserving localization accuracy.

2.2 Machine-Learning and Computer-Vision Algorithms

Machine learning plays a crucial role in the localization of autonomous vehicles by processing data from different sensors and maps in order to determine the vehicle's position more accurately. Deep learning techniques, such as Convolutional Neural Networks (CNNs), Support Vector Machines (SVMs), and Long Short-Term Memory (LSTM) networks, are used to process sensor data and improve localization accuracy [CE24].

Localization also relies on precise positioning data, which can be obtained through GPS and other positioning systems. The lateral control algorithm uses GNSS localization data and maps to guide the vehicle along the correct path, while the longitudinal control algorithm uses inter-vehicle gap measurements to maintain safe distances [KT01].

For instance, LSTM-based methods have been used to learn uncertainties in wheel speed measurements and tires, enhancing the robustness of localization against GNSS outages [LSY+22, SMM23]. These methods enable to predict the position and angle of a vehicle over long sequences with greater accuracy compared to traditional methods such as the Extended Kalman Filter (EKF), however by combining both allows one to increase accuracy even more [TC24]. Moreover, machine learning algorithms can help in lane detection, a critical component of localization. Models like RESA [ZFZ+21], PINet[6], and CondLaneNet [LCZT23] have achieved high accuracy and F1 scores in lane detection without the need for extra training data.

Another important technique for better localization is computer-vision algorithms, which enable autonomous vehicles to detect and understand their surroundings accurately. These algorithms use cameras and various sensors such as LiDAR and radar to capture data from the environment, which is then processed to identify objects, road lanes, and other critical elements necessary for safe navigation [SA23].

Additionally, object detection algorithms like YOLO (You Only Look Once) [RDGF16] and Deep SORT (Simple Online and Realtime Tracking) [WBP17] are employed to identify specific objects in real-time. These algorithms are essential for recognizing and tracking vehicles, pedestrians, and other dynamic elements in the vehicle's path. [KPP+21]

One of the key techniques used in localization is semantic segmentation, which involves labeling each pixel in an image with a category such as road lanes, vehicles, pedestrians, and obstacles. This helps the vehicle to distinguish between different elements in its environment and make safer decisions [KPP+21].

2.3 Particle filters

Particle filters, also known as Sequential Monte Carlo (SMC) methods, are powerful tools for solving the localization problem in robotics, especially under uncertainty. They

[6]PINet GitHub, https://github.com/koyeongmin/PINet

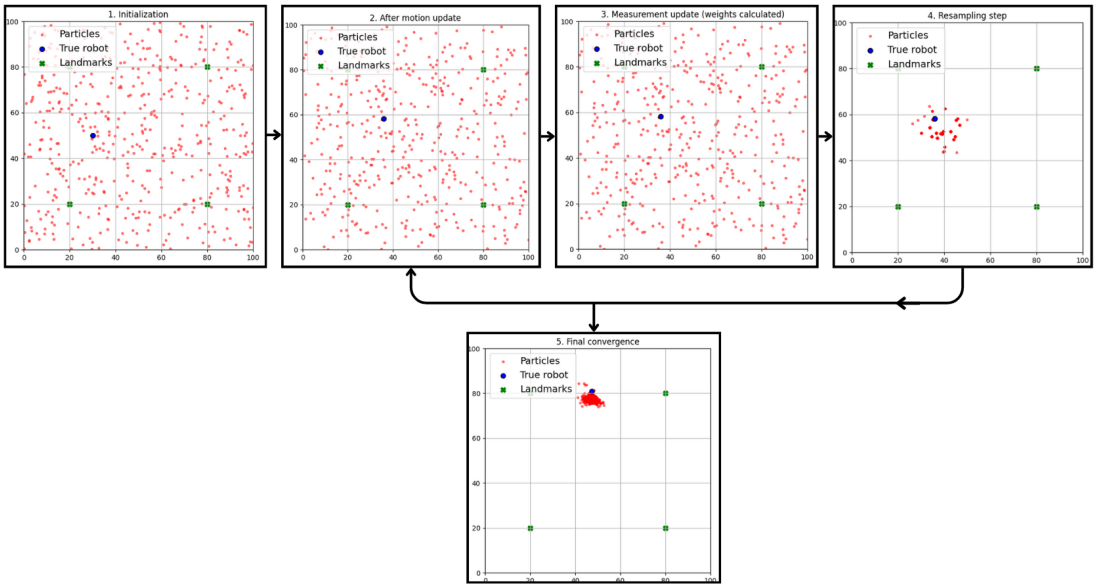


Figure 4. The steps of a particle filter. Starting from left: (1) initialization (particles scattered across the world), (2) motion update(update based on robot's movement), (3) measurement update (update based on robot sensing landmarks), (4) resampling particles based on weights, and (5) convergence (particles cluster near the true position). Steps 2 to 4 are repeated over multiple iterations until convergence is achieved.

operate by representing the probability distribution of a robot's state using a set of discrete samples, or particles, each associated with a weight that reflects its likelihood. As the robot moves and collects observations, these particles are updated through a cycle of prediction, correction, and resampling (see Figure 4).

A particle filter operates by maintaining a set of weighted samples, known as particles, to represent the probability distribution of a system's state. These particles collectively approximate the posterior distribution, allowing the filter to estimate the most likely state.

According to Thrun et al. [TF05], the particle filter process is composed of the following main steps: initialization, prediction, correction and resampling. At the start, particles are distributed across the state space, either uniformly or based on prior information. For vehicle localization, this could mean assigning random poses (position and orientation) to particles within a bounded region of a map. Each particle's state is updated using a motion model that incorporates vehicle control inputs (e.g., steering, velocity) and stochastic noise to account for uncertainty. This step predicts the vehicle's new pose based on its movement.

Observations, such as sensor data from cameras or LiDAR, are compared against expected measurements generated by each particle's hypothesis. The weight of each

particle is then updated to reflect the likelihood of its state given the observed data. For instance, in map-based localization, the alignment of particle poses with map features significantly influences their weights. The process iterates as new observations and control inputs are received, refining the state estimate over time.

The flexibility of particle filters allows them to handle multi-modal distributions, making them particularly useful in scenarios where the robot's initial position might be ambiguous. Thrun et al. [TF05] emphasize the adaptability of particle filters to varying environmental conditions, highlighting their effectiveness in GPS-denied scenarios where alternative data sources, like visual or LiDAR information, must be utilized for localization.

2.4 Open Street Map

Open Street Map (OSM) [Wik25] is a free, collaborative, and editable map of the world, created and maintained by a community of volunteers. Unlike proprietary mapping platforms, OSM is built largely from scratch and released under an open database license, allowing free or nearly free access to its map images and the underlying map data. Being open-access has made OSM a versatile resource for a wide range of applications, from individual projects to governmental and commercial uses.

Over the years, OSM has achieved significant milestones in mapping coverage and accuracy becoming a viable alternative to other map providers in many countries and for various use cases. However, as a dynamic project, the map is constantly evolving, with contributors working to improve and expand its scope.

OSM is a global community effort that involves active fieldwork - volunteers survey towns and countryside to collect geographic data, which is then collaboratively edited using wiki-style software. The maps grow and improve continuously through the contributions of users who can upload GPS tracks, derive data from aerial imagery (such as Bing), or use other mapping techniques. [Wik25]

3 Related work

Vehicle localization has been extensively studied, with numerous approaches relying on the availability of Global Navigation Satellite System (GNSS) signals and communication with other vehicles or infrastructure. These systems are generally designed for conditions where GNSS is available, although sometimes inaccurate [LBAN19], or where vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communication is used to enhance the localization accuracy. Despite their effectiveness, such assumptions limit their applicability in GNSS-denied or communication-constrained environments. Moreover, for true autonomy, a future level-5 vehicle should still be able to localize itself in case of a GNSS sensor failure. In this chapter, an overview and analysis of existing works is given, discussing their strengths and weaknesses.

3.1 Monte Carlo Localization

As briefly discussed in Section 2.3, one approach to vehicle localization is Monte Carlo Localization (MCL) methods, where a rough initial GNSS position can be used to localize the vehicle. These methods typically rely on a combination of visual odometry (VO) and map matching, where the vehicle's motion model is updated based on sensor data, and hypotheses are tested against preloaded maps.

A representative approach, described in the work of [FL13], uses MCL with visual observations as input and chamfer matching to align vehicle hypotheses with the map. In this framework VO output is fed to the motion model of particle filtering algorithm evaluation according to its alignment to the map. However, this method assumes the presence of an initial GNSS position, which can be several hundred meters off, and typically works in environments where GNSS is unreliable but still present.

Yan et al. [YS19] proposed a global localization algorithm using a 4-bit semantic descriptor within a particle filter framework, relying on building and road information extracted from OSM. This approach is computationally efficient, as the descriptor simplifies comparisons while requiring minimal memory.

Challita et al. [CB09] demonstrated a centralized nonlinear filter to enhance vehicle positioning accuracy in urban scenarios. While the reliance on communication and GNSS makes it unsuitable for completely GNSS-denied settings, it showcases the benefits of integrating diverse data sources, such as GNSS, vision, and communication between vehicles for localization.

Akai [Aka22] presented a novel graphical model and formulated the Bayesian filtering for the simultaneous localization, sensor measurement class estimation, and reliability estimation problem. They used Rao-Blackwellized particle filter [ADR00] to implement the simultaneous estimation system. proposed a method to generate new hypotheses in the form of particle filters competing with each other to be selected as the system output. Their work allowed the robot to be unaware of its initial position.

3.2 Machine learning methods

Another recent approach introduces deep learning to localization, using visual observations and OSM data to perform localization. OrienterNet [SB23] proposed a deep neural network that estimates the pose of a query image by matching a neural bird-eye view (BEV) with available maps from OSM. Instead of focusing on specific classes, this method managed to produce sub-meter accuracy using all semantic classes available in OSM.

Bârsan et al. [BWPU20] introduced a method to achieve centimeter-level accuracy by using a convolutional neural network to project online LiDAR sweeps and intensity map into joint embedding space. Their work also takes the availability of an inaccurate GNSS position for granted.

Wei et al. [WU19] focused on the problem of HD maps used in autonomous driving, introducing a task-specific map compression framework optimized for localization. They proposed a fully convolutional network that learns to binarize the map features and compresses it using run-length encoding on top of Huffman coding. OSMLoc [LCK+24], on the other hand, integrates the Depth Anything model [YKH+24] for geometric guidance and semantic information from OSM data to create a brain-inspired single image-to-OSM (I2O) visual localization framework. It uses the DINOv2 [ODM+24] encoder to extract image features and incorporates geometric information into camera-to-BEV transformation. The framework of MapLocNet [WZL+24] is built on a transformer-based hierarchical feature registration method. It uses EfficientNet to encode surround-view images into BEV space and processes navigation maps using a U-Net [Ron15] architecture.

3.3 Using vision for localization

In urban environments, the reliance on 3D building information has been explored to improve localization accuracy. A method proposed by Ballardini et al. [BS16] combines stereo image streams from in-vehicle cameras with a 3D reconstruction pipeline to detect building outlines, which are then matched to cartographic knowledge provided by OSM to enhance localization. While effective in GNSS-denied urban areas producing a lane-level accuracy, this method assumes that some form of map data and prior knowledge of building appearances are available, which may not be the case in all environments.

Zhou et al. [ZCSA+21] proposed a method using an observation model and motion model that were integrated with a particle filter. With this approach they managed to link map tiles to ground level images. Miller et al. [MCH11] used traditional bootstrap particle filter in their approach to fuse satellite-based localization information with vision-based road information. They also introduced recursive particle filtering and mentioned experimentation with extended GNSS blackouts.

Pepperell et al. [PCM16] proposed a robust place recognition framework for vehicles and robots using a directed graph representation of branching of contiguous sections of

imagery at intersections alongside with a particle filter to explore these paths.

A different approach represented in [BGU16] that had no knowledge of the vehicle beyond the gross region mapped the vehicle trajectory based on the VO of the OSM data. Despite getting an accuracy of 4 meters after 52 seconds of driving their method suffered noticeably in straight segments of the road where no recognizable turns were detected.

3.4 Street view text and sign detection

One helpful technique to improve localization accuracy is the detection of map features. Street-level panoramic images have been exploited for large-scale traffic sign localization and classification, improving localization accuracy and giving scene context to self-driving vehicles. A learning-based system by Hazelhoff et al. [HW14] detects, positions, and classifies road signs from non-dense images, achieving high localization and classification accuracy. While this system is geared toward traffic sign inventory management, its integration of visual data with environmental models highlights the utility of leveraging street-level imagery for broader localization tasks.

Similarly, the detection can be broken into even smaller parts by detecting texts in character level. For example [ZTD+22] created a multi-segmentation text detection (MSTD) network for character-level street view text detection, which had broader receptive field. It improved previously proposed sequence-to-sequence methods by requiring less training data, being explainable and improving the performance of character recognition and localization.

Equally important to detecting and classifying road objects is mapping them accurately to maps. This task was proposed by Zhang et al. [ZL21] who detected in their work road objects and used an attributed topological binary tree (ATBT) based on urban rules to depict relations of topologies, attributes, and semantics to match them with OSM map features.

Many works have focused on providing quality datasets in order to improve autonomous vehicle systems. One such example would be the work of Zhang et al. [ZDP+21] creating a benchmark dataset for urban scene text recognition, which could be used to improve localization accuracy.

3.5 Motivation

While the discussed methods demonstrate significant progress, they share common limitations mostly relying on GNSS signals or external communication infrastructure. These dependencies make them ineffective in environments where GNSS is entirely unavailable or where external infrastructure cannot be relied upon. Moreover, many methods assume detailed prior environmental knowledge, such as high-definition maps, pre-labeled landmarks, or previously scanned road geometries, making them unsuitable for localization in previously unseen areas.

This gap in the literature motivates the current work, which investigates vehicle localization using particle filters and image detection assuming no availability of GNSS or vehicle-to-infrastructure communication. This research aims at enhancing localization techniques that can work in completely GNSS-denied environments, providing solutions that are not reliant on pre-built environmental representations, annotated map data, or external communication.

4 Methodology

This section will focus on experimental setup and code-related information, discusses different evaluation metrics, and gives an overview of the tools used.

4.1 Basic assumptions

In order to get the particle filter working as needed, some basic assumptions had to be introduced to the system:

- **The autonomous vehicle will only navigate on roads.** Since autonomous vehicles will only drive on roads, the framework can be simplified by initializing particles only close to the roads.
- **There is no previous knowledge of GNSS location.** However, the general region where the car is located is known. To account for this, a city-scale localization is used - specifically, the car is assumed to be within a 20-kilometer radius from the city center.
- **The car will detect only street signs and bus stop signs.** In this framework, only the street names and bus stop names are used as map features for localization.

4.2 Proposed approach

There were two main components to consider when building the proposed framework: the vision-based detection model and the particle filter. Initially, the focus was mostly on the particle filter, so the detection model was replaced with predefined street sign locations that were collected using Google Maps. The proposed framework is depicted in Figure 5. It was decided to implement missing parts later when time was left, but due to time constraints, unfortunately the vision-based detection model was not implemented.

Here, a description of the proposed framework is given:

1. For inputs, the start and end coordinates of chosen path were used starting from Lille street and ending in Tähe street in Tartu. Then OSM data, which included shortest path generated from given start and end coordinates, and landmarks (street name locations collected based on ground truth landmarks). Finally, ground truth landmarks were used, which were collected during observations using Google Maps.
2. The motion planner was used to generate motion commands for vehicle and particle movements as well. For this, motion distance and heading were calculated using Euclidean distance and arctangens with small added noise. Then these motion commands were used to move the vehicle and particles in each particle filter step.

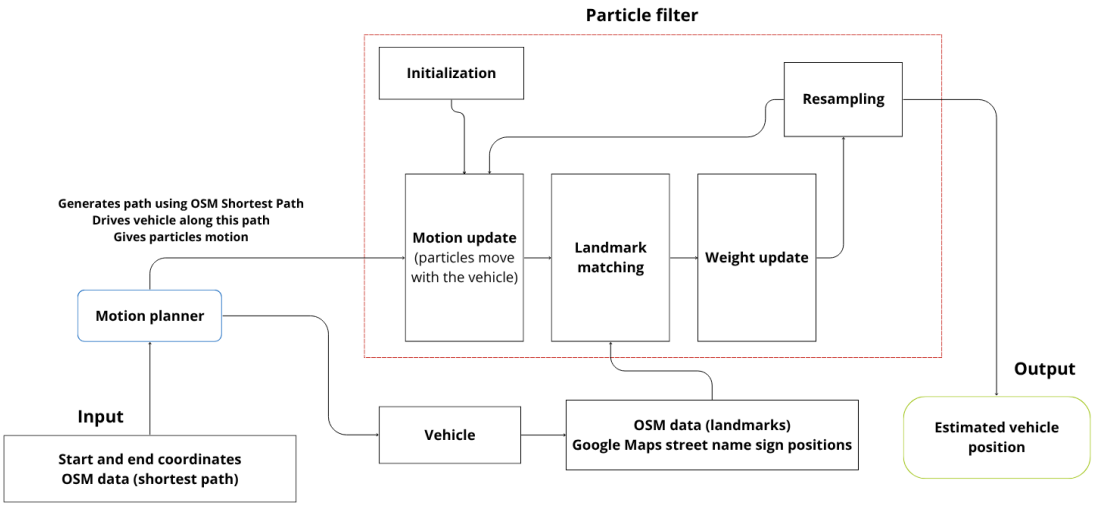


Figure 5. Diagram of proposed framework without separate street sign detection model.

3. Particle filter included multiple steps starting with initialization of the particles along the road network with an offset to the right side. Continuing with the motion update step, where motion commands were fed to the particle filter. Then, landmark matching was performed comparing particle positions with ground-truth landmarks and OSM landmarks. Based on ladmark matches particle weights were recalculated and finally, the last step was the resampling of the particles. After resampling, an estimated vehicle location was calculated taking the average location of particles. These steps (excluding initialization of particles) were performed in multiple iterations until the vehicle was no longer moving.

4. The output of this framework was an estimated location of the vehicle. When collecting these positions between each iteration, an estimated vehicle movement path could be reconstructed.

4.3 Experimental setup

The codebase was fully built using Python programming language. The experimental part was divided into two main sections: exploring vision-based detection on street-view images and creating workflow of a particle filter.

4.3.1 Particle filter setup

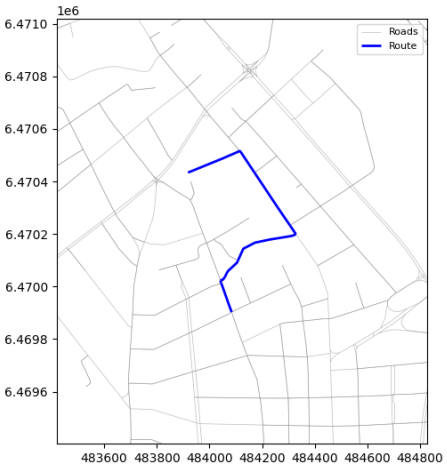
A particle filter was implemented, which is a recursive Bayesian filter that uses a set of weighted samples (particles) to represent the probability distribution of the system's

state (see Algorithm 1). For testing purposes the initial data was simulated from specific route that was previously selected starting from Lille street and ending at Tähe street in the city of Tartu (see Figure 6). These coordinates were collected from OSM and the shortest path was then calculated. This path was used purely for testing and not for the localization itself. From the route coordinates the motion commands were extracted using Euclidean distance for distance and arctangens for rotation or heading. These motion commands were used to simulate vehicle's movement and to update particle filter step as well. After that, the particles were initialized close to the road network because the vehicle can possibly drive only on the roads (this was one of the assumptions given in previous section) (see Figure 6).

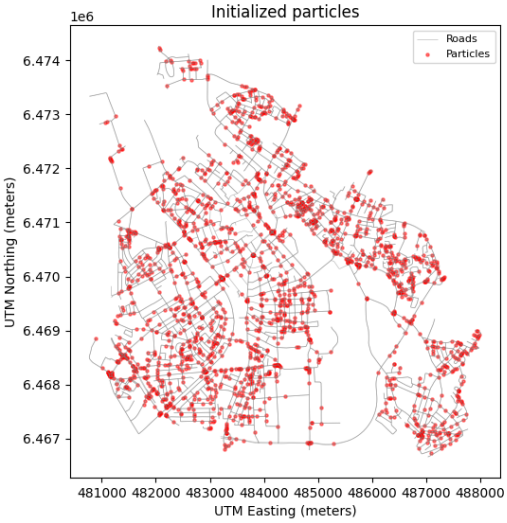
Algorithm 1: An algorithm of a Particle Filter

Data: $N, \quad u_t, \quad z_t, \quad x_{t-1}^{[i]}, \quad w_{t-1}^{[i]} \quad \text{for} \quad i = 1 \rightarrow N;$
/* Number of particles, control input at time t, observation at time t, particle set from previous time step */
Result: $x_t^{[i]}, w_t^{[i]} \text{for} i = 1 \rightarrow N;$ /* Updated particle set */
for $i = 1 \rightarrow N;$ /* Resampling part */
do
  $x_t^{[i]} \approx p(x_t | x_{t-1}^{[i]}, u_t);$ /* Motion model with noise */
  $w_t^{[i]} \leftarrow p(z_t | x_t^{[i]});$ /* Evaluating likelihood */
for $i = 1 \rightarrow N$; /* Normalizing weights */
do
  $w_t^{[i]} \leftarrow w_t^{[i]} / \sum_{j=1}^{N} w_t^{[j]};$
$N_{eff} \leftarrow 1 / \sum_{i=1}^{N} (w_t^{[i]})^2;$ /* Effective number of particles */
if $N_{eff} <$ *threshold;* /* Resampling part */
then
  Draw N new particles from $x_t^{[i]}$ with probability $w_t^{[i]}$;
  Set all $w_t^{[i]} \leftarrow 1/N;$

The main "loop" of particle filter was initialized with 2000 particles to balance computational efficiency and localization accuracy, and a testing vehicle which imitates real life vehicle. For landmark simulation, a set of street names were artificially introduced to the framework based on Google Maps manual observations. Based on these selected/sensed landmarks, new weights were calculated for the particles that were then resampled, counting only the highest weights. The particles had to move exactly the same as the vehicle in order to approximate its location. During prediction step each particle was moved based on the control inputs including velocity, turning angle and noise. A simple



(a) Route chosen for the experiment starting from Lille street and ending in Tähe street in Tartu.



(b) Particles initialized close to the roads.

Figure 6. Two figures showing the selected route and initialized particles.

motion model was used, adding Gaussian noise to simulate real-world uncertainty.

$$x' = x + v \cdot \Delta t \cdot cos(\theta + \delta)$$

$$y' = y + v \cdot \Delta t \cdot sin(\theta + \delta)$$

$$\theta' = \theta + w \cdot \Delta t$$

where v is the linear velocity,w is the angular velocity, and δ is the random noise.

Each particle's weight was updated based on how well the expected sensor readings from that particle's position match the actual measurements. For this, a likelihood function was used comparing the predicted sensor readings (distance to street signs) to the actual measurements, applying a Gaussian error model. Particles were resampled according to their weights to focus on more probable states. Particles with higher weights should be duplicated, whereas those with lower weights get discarded, ensuring that computational resources focus on the most probable regions of the state space. Systematic resampling was implemented to avoid degeneracy, where a small number of particles dominate the population. Finally, the estimated position of the vehicle was computed as the weighted average of all particles.

Different approaches were tried during experimentation: filtering out landmarks based on vehicle's estimated orientation received from motion commands (only landmarks in front of the vehicle), filtering landmarks based on their distance to the estimated location

of the vehicle (200 meters), snapping the vehicle onto the closest road each time the movement was updated (finding the closest road point and assigning its value to the vehicle), and different amount of particles initialized. The calculation for landmark filtering is presented below:

$$angle\_diff = (angle\_to\_landmark - robot\_orientation + \pi)\%(2 \cdot \pi) - \pi$$

The resulting difference had to be less than 90 degrees for the landmark to be in front of the vehicle. Finally, some additional experimentation was made using false sensing data and out-of-scope (outside of Tartu) location data.

4.3.2 Vision-based detection on street-view images

All street names (referred to as landmarks) that ideally would be detected by our vision-based detection model were artificially fed into the particle filter. At first, a route was chosen for the experiment which was then drove through using Google Maps application (see Figure 7). All of the visible landmarks and locations were recorded and saved in format

$$[name, POINT(x, y) + \theta, bool],$$

where $name$ is the detected street name, $POINT(x, y)$ is the location as shapely geometry type that represents a single coordinate with a small added noise θ, and $bool$ is the boolean value of the sign being a bus stop. Additional landmarks were also collected from OSM where, given a ground truth landmark collected from Google Maps, all of the matched street name locations were considered as detected landmarks.

By assuming that our detection model works correctly we can fetch street names or bus stops and get an estimation of the vehicle's position. For that, all possible coordinates of a given street are fetched from OSM and given as input landmarks for the particle filter. To get the estimated location of vehicle, the following steps are executed:

- Road information is fetched based on the given street name that the vehicle is expected to detect using OSM data.
- Using the fact that the robot should be located on the detected street, the robot's position is initialized using the midpoint of the street curve (by length), ensuring a more realistic and representative location. A small amount of noise is added to introduce robustness.
- Finally, calculated location estimate is then snapped to the closest road point. This step ensures that the vehicle is moving on the road (see Figure 8).

Figure 7. Example frame from Google Maps moving along Kalevi street, where the visible street sign is marked with red box.

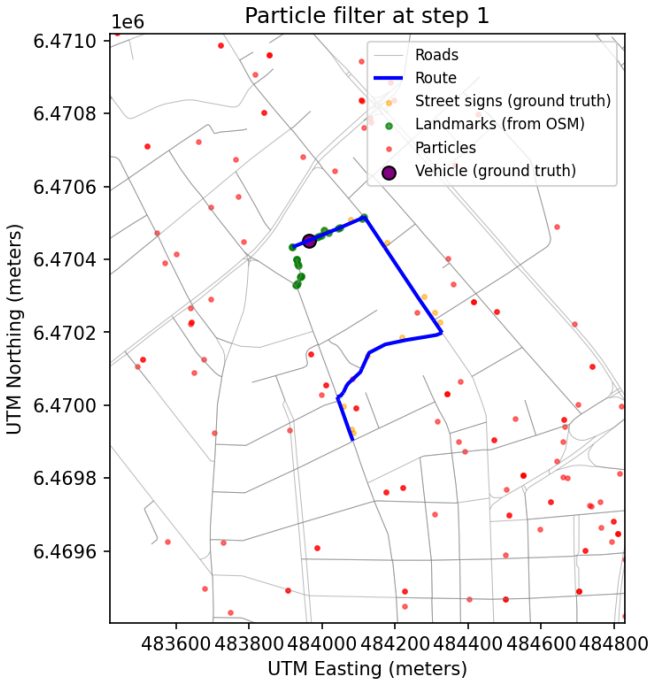


Figure 8. The vehicle's ground truth location (purple dot) after fetching the ground truth landmark (street name), landmark information from OSM and snapping the vehicle onto the closest road point.

For each landmark detected, the previously introduced steps were executed with a difference of finding the road section's middle point. This time the vehicle's movement was updated based on the motion model, and the actual coordinates of additional landmarks from OSM were fed into the particle filter.

4.4 Evaluation metrics

A meter-level error was calculated to evaluate the particle filter's work and accuracy. The ground truth landmarks were taken from Google Maps street view as seen in Figure 7 and the ground truth route was collected from OSM where given the starting and end point a shortest path was then calculated with all coordinates of the path. These coordinates were then used to calculate motion commands to make the vehicle move as it would in real life, calculating distance and heading. For the purpose of making visualization look consistent, the full distance of the route was divided into 50 meter sections.

4.5 Tools

For map related work, the OSMNX library was used, as well as shapely and geopandas. For writing the manuscript, Overleaf was used. GitHub was used to store and share all of the experimentation and framework code.

5 Results

In this chapter, the results of the experiments introduced in Section 4.3.1 are presented, then some additional experiments are shown, and finally an overview of the evaluation approaches is given.

5.1 Experiment with initial framework

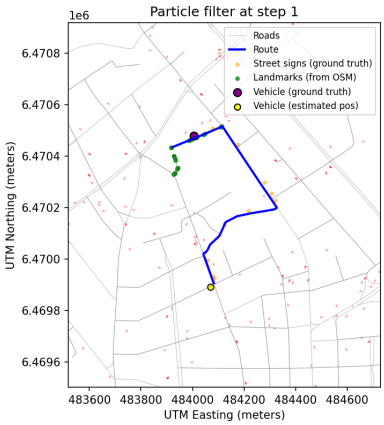
The first test run was performed on the initial framework with no further improvements. The vehicle's position was initialized on a random road node, after which an artificial sensing was given to the vehicle (detected landmark). Based on that, an estimated improved location was set. During each particle filter step, the vehicle received additional motion commands, more landmarks, the particle weights were updated based on these landmarks, and a new set of particles was resampled.

There was no additional filtering on particles or landmarks considering vehicle's rotation and the distance to the landmarks. The vehicle was also not snapped to the road network, which created situations where the vehicle was located outside of the road network (not strictly on the roads). The particle filter was able to localize the test vehicle with an error of 220 meters. From Figure 9 it is visible that the particle filter is able to converge during 20th step and get relatively close to the vehicle's actual position but being still off more than 200 meters with an average error of 456.52 meters. Green dots visible in the figure are all sensed landmarks that are with a maximum distance of 200 meters from the vehicle.

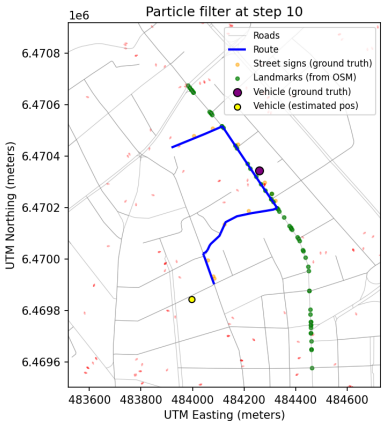
5.2 Experiments with landmark filtering

The second test run had a similar setup with some major improvements to accuracy in localization, reducing error from 456 meters to 54 meters. Firstly, the vehicle's sensing area was reduced to only the front part. While previously possible landmark locations were scattered across the street, including behind the vehicle, it was now considered only in front of the vehicle positions. Secondly, the distance of sensed landmark locations was reduced to 100 meters to avoid including long streets when calculating particle weights, as seen on Figure 9(b) and Figure 10(b) (green dots). For instance, in some cases there were around 200 possible locations for sensed landmarks but only around 20 of them were close to the vehicle (see Figure 10).

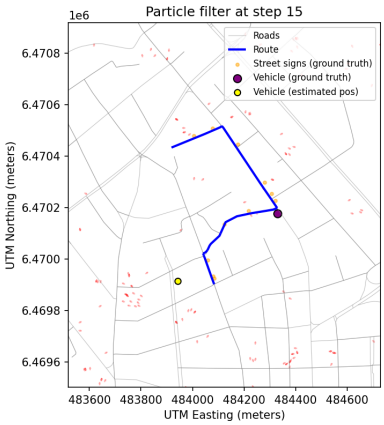
Finally, after updating vehicle's location it was then snapped to the closest road point which in some cases greatly improved accuracy but in some cases introduced false positives. For example, instead of choosing the road in front of the vehicle, the right-hand road was chosen (see Figure 11). One more experiment was conducted where the previous road information was used when snapping the vehicle to the road during



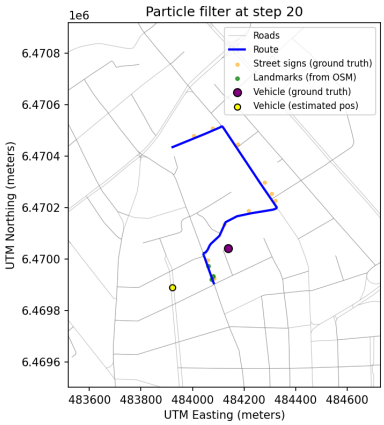
(a) Particle filter step 1.



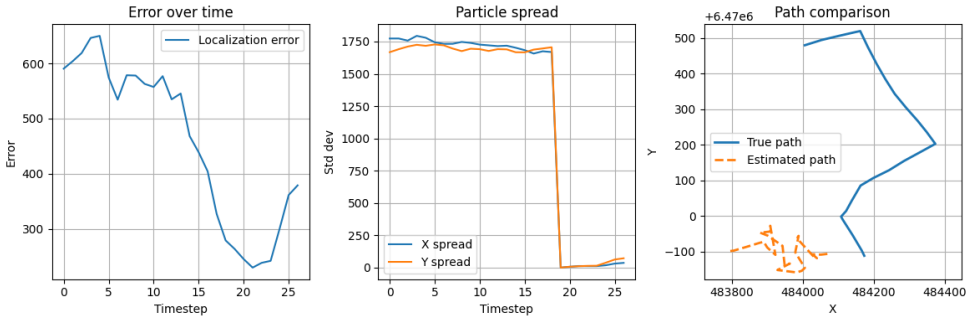
(b) Particle filter step 10.



(c) Particle filter step 15.

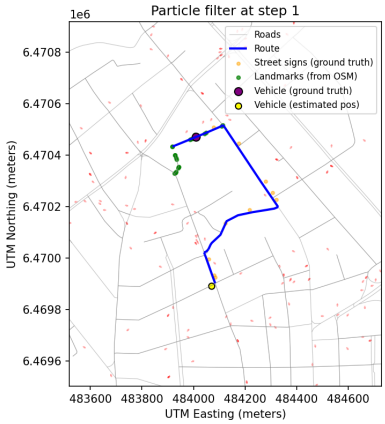


(d) Particle filter step 20.

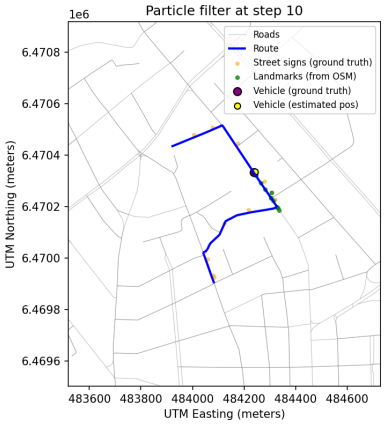


(e) Evaluation metrics: error over time, standard deviation of particle coordinates, and comparison of ground truth and estimated path.

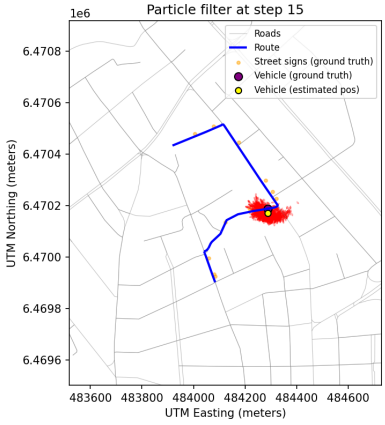
Figure 9. First experimentation results where path is given as a blue line, vehicle's ground truth location as purple dot, vehicle's estimated location as yellow dot, particles as red dots, sensed landmarks taken from OSM as green dots and ground truth landmarks collected from Google Maps as orange dots.



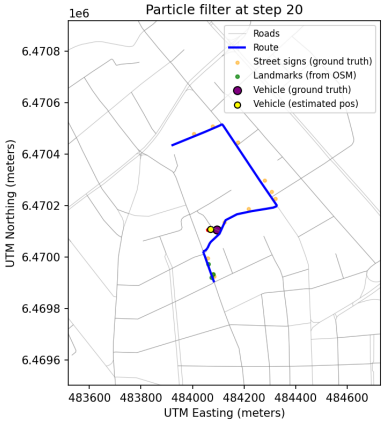
(a) Particle filter step 1.



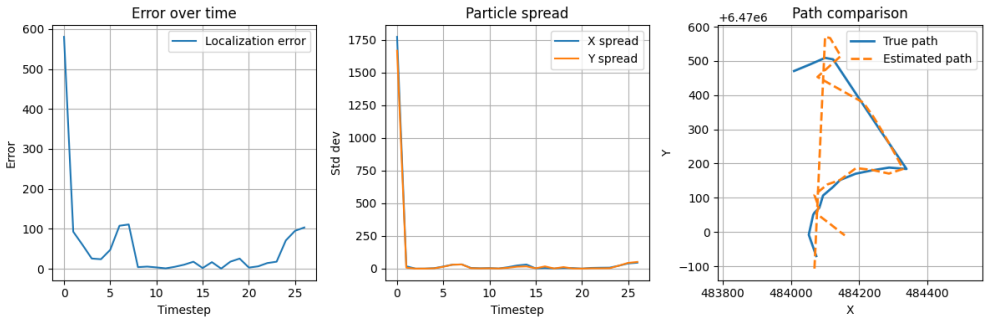
(b) Particle filter step 10.



(c) Particle filter step 15.



(d) Particle filter step 20.



(e) Evaluation metrics: error over time, standard deviation of particle coordinates, and comparison of ground truth and estimated path.

Figure 10. Second experimentation results where vehicle is given as a purple dot, ground truth landmarks as light blue dots, particles as red dots and sensed landmarks taken from OSM as green dots.

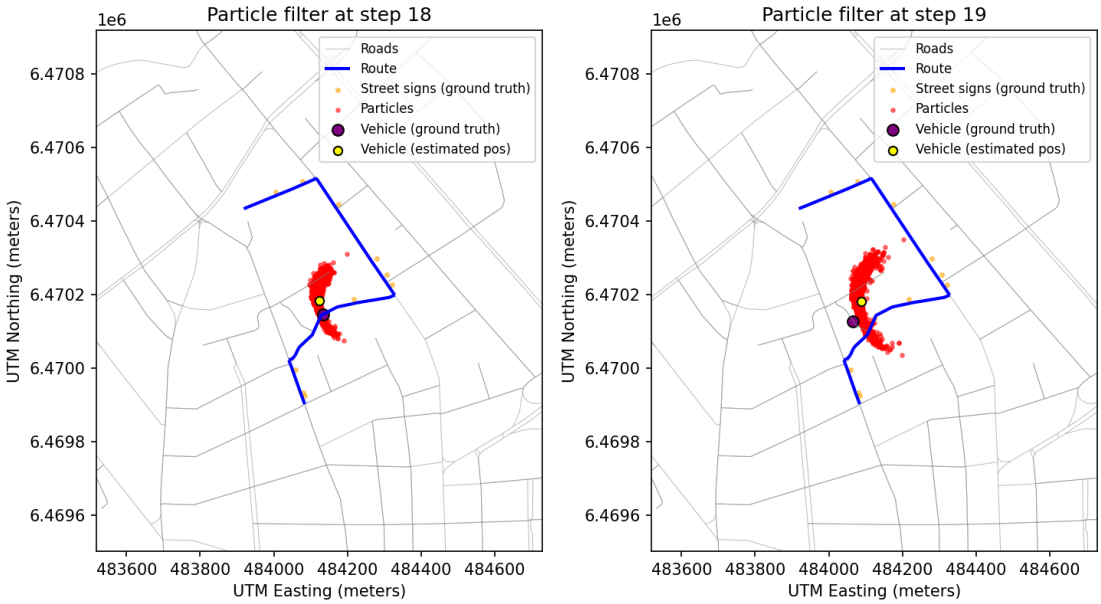


Figure 11. Example of a situation where incorrect road is chosen for the vehicle's location.

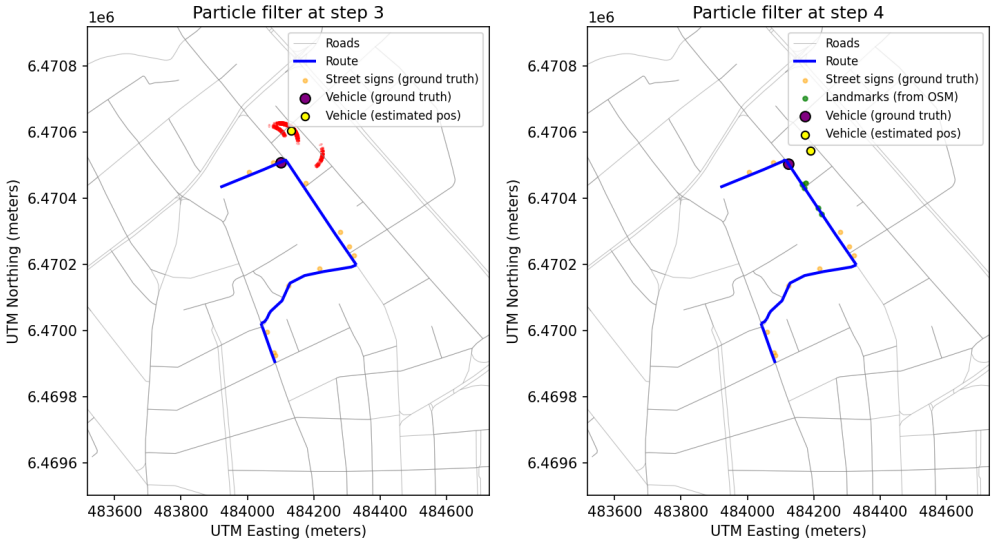
the next particle filter step, but this did not improve accuracy even though eliminating "jumping" between road segments. For more details see GIF[7].

5.3 Experiments with corrupted data

Couple of experiments were conducted by inserting incorrect or slightly corrupted data into the system, more specifically wrong street names. This was done to evaluate system's work when encountering unexpected data. When changing only one of the names that the vehicle should detect, the results improved even more in some cases (see figure). Additional information is visible in GIF[8]. For example, using 2000 particles, changing one landmark Kalevi street to Turu street in Tartu, the error reduced from 54 meters to 53 meters.

One interesting observation was that changing up to three landmarks resulted in even lower error rate, 52 meters, but the main reason for this was the false information that was chosen. Using Turu street as the false landmark - which is relatively close to the actual path the vehicle is driving - increased accuracy because Turu street is longer and therefore has more landmarks (see Figure 12). The introduction of false data did not add much divergence into the particle filter because all additional landmarks that were

[7]https://tinyurl.com/23nss8b6
[8]https://tinyurl.com/42dume9t



(a) Particle filter step 3.

(b) Particles filter step 4.

Figure 12. In the left figure particles (red dots) start moving closer to the Turu street but in the right figure the correct landmark is detected (green dots) and the particles start moving back towards vehicle's position.

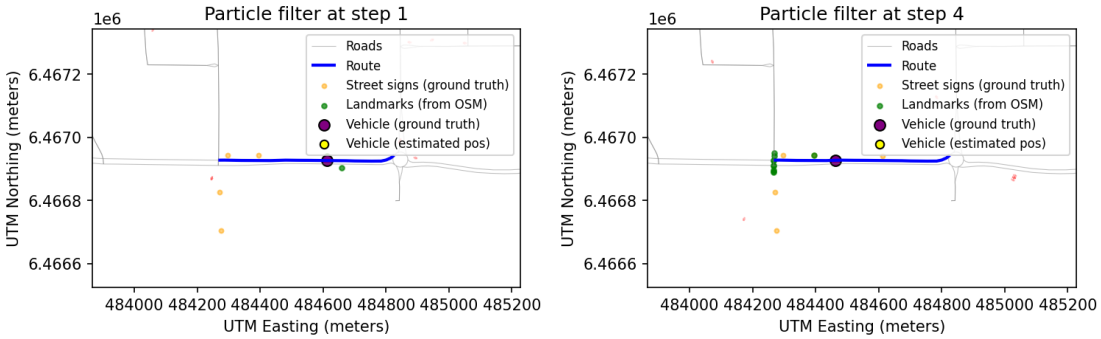
collected from OSM went through a filter and therefore landmarks too far away from the vehicle get filtered out.

An experiment was carried out using detected data outside the city of Tartu (outside of map boundaries). Since the code was snapping the vehicle onto the closest road section, when the vehicle went outside of the map, it was still snapped inside the map and the location started to diverge from the ground truth, as seen in the Figure 13. Additional information found in GIF[9]. From Figure 13, we can see that two of the sensed landmarks are outside of map boundaries (orange dots). Interestingly, the landmarks collected from OSM (green dots) had correct coordinates and were visualized in correct places, but the road network was missing that section, which is why the vehicle could not move to the left and got snapped to the current road section.

5.4 Evaluation of the experiments

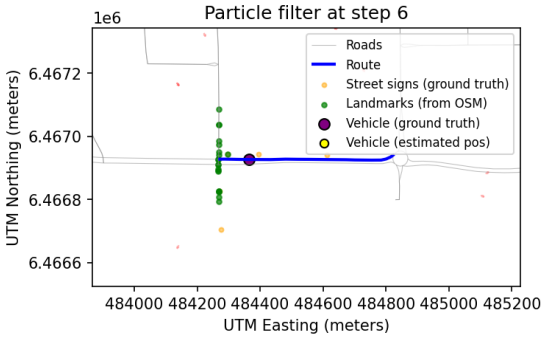
For evaluating framework's accuracy, a localization error was recorded over time, particle spread, and path comparison, for example, using 2500 particles (see Figure 14). These evaluation metrics were recorded for each experiment introduced in Section 5. The estimated path was calculated based on particle's average position in each step. As seen

[9]https://tinyurl.com/mv9eb25n

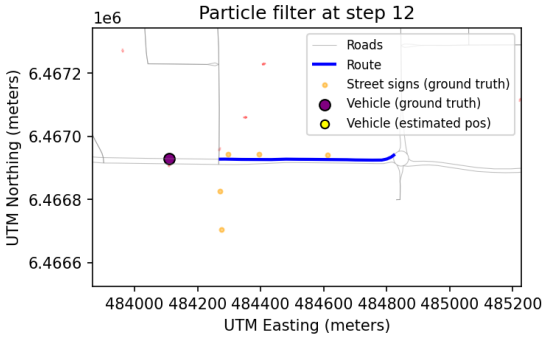


(a) Particle filter step 1

(b) Particle filter step 4



(c) Particle filter step 6



(d) Particle filter step 12

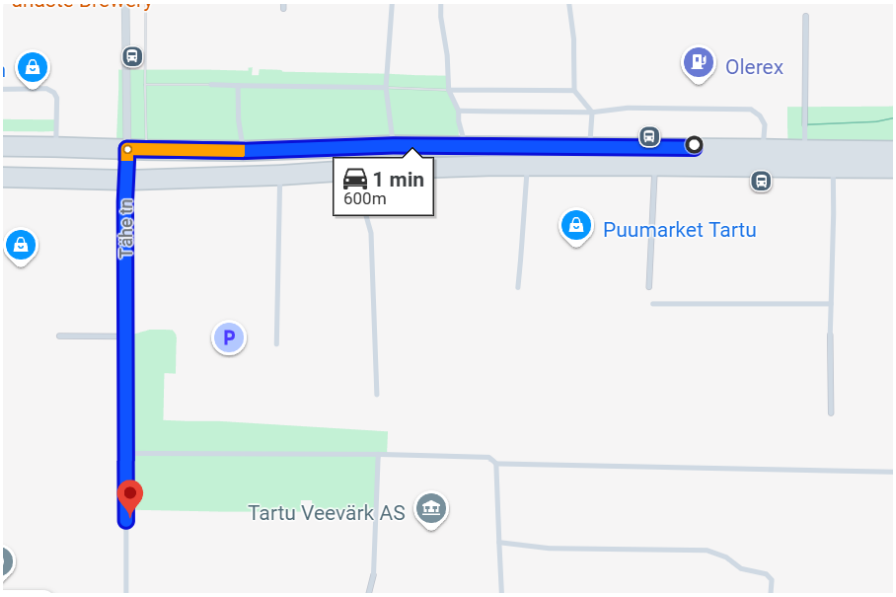


Figure 13. Example of a situation where the vehicle starts moving outside of the bounds of the map but gets snapped onto the closest road. Orange dots are detected (ground truth) landmarks, green dots are additional OSM landmarks, purple dot is the vehicle ground truth location and red dots are particles.

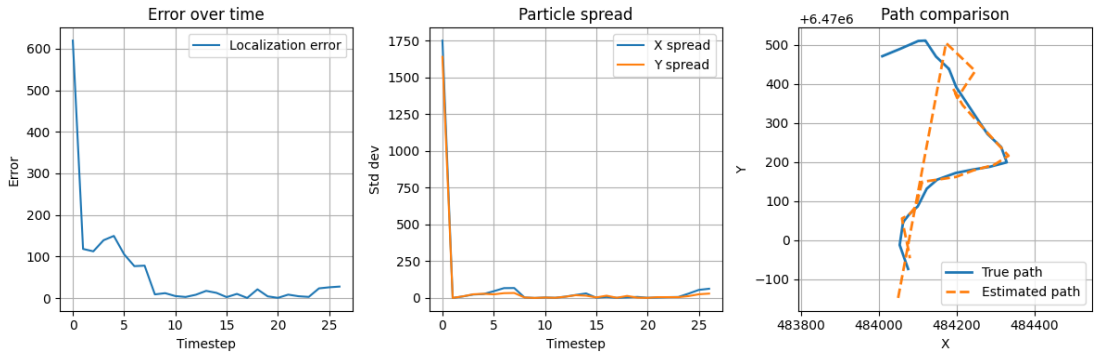


Figure 14. Evaluation results of the particle filter with 2500 initialized particles, filtering of landmarks and particles, and snapping the vehicle onto the closest road section. Starting from left: particle filter error over time, particle spread over time (x and y coordinates), and vehicle's movement path (blue line) vs estimated path (orange dashed line) from particle filter.

in the figure, the error increases over time due to the scarcity of landmarks. This means that the particles start spreading uniformly farther away from the vehicle.

Additional testing was conducted, including the landmark filtering introduced in Section 5.2, in order to compare different amounts of particles used to find the best possible set (see Figure 15). For that, a set of particle counts, from 500 - 4000, was initialized and looped over time to collect average errors. Each particle count was executed 20 times and the average error was then calculated. The execution time compared to the amount of particles was also recorded (see Figure 15).

From these two figures it is possible to determine the best possible amount of particles, which would be 2750 since it has relatively low error rate and a short execution time compared to 3500, which, although having the smallest error rate, has roughly 1.6 times longer execution time. Another great error and execution time balance would be received using 2500 or 3000 particles.

After changing the particle amount to 2750, the average error was higher (250.27) than with the particle amount 2500. Therefore, another test round was performed with 2500 initialized particles, as seen in GIF[10], and the average error was 59 meters (see Figure 16). It was decided to conduct one more experiment with a higher particle count, more specifically 3000 particles, which yielded even better results, as visible in Figure 17. The results of experimentation with different amounts of particles are presented in Table 1.

[10]https://tinyurl.com/2un2hfcx

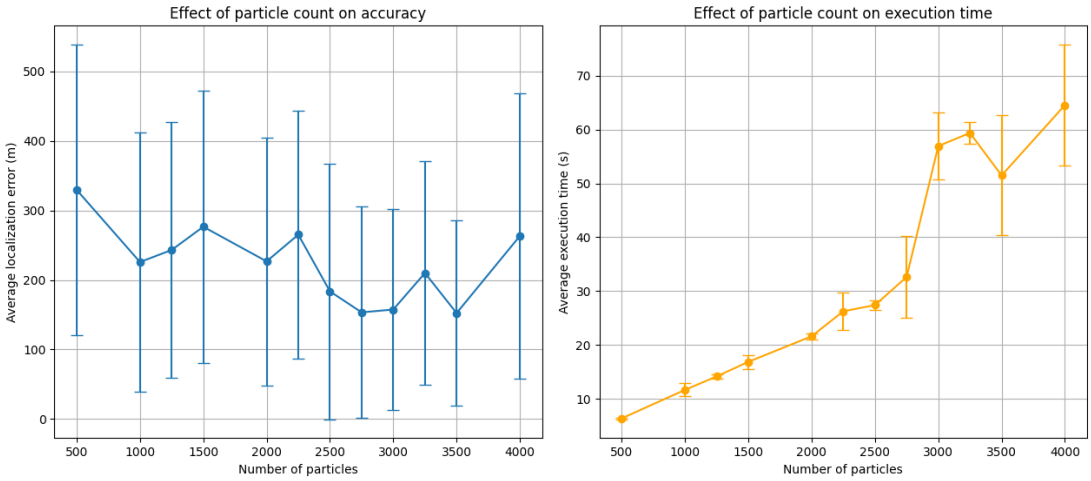


Figure 15. Effect of the amount of particles on accuracy and execution time. For each particle count the experiment was run 20 times; error bars represent standard deviation.

5.5 Discussion

As seen from the results section, the framework works surprisingly well given the lack of positional information and the absence of GNSS location. As expected, the particles start diverging from the vehicle as soon as there is a shortage of sensed landmarks. However, as soon as a street name is detected, the particles get new weights assigned and start clustering in probable vehicle locations.

5.5.1 Discussion on experiments with landmark filtering

In Section 4.3.1, different approaches for experimentation were defined. Although some changes did not add any value to the testing part, most of them improved localization accuracy (see Table 2). For example, filtering out landmarks based on their distance to the estimated location of the vehicle (200 meters) managed to improve localization accuracy by reducing error from 361.96 meters to 295.44 meters, whereas filtering based on vehicle's estimated orientation did not improve the localization accuracy. As expected, filtering landmarks based on distance significantly reduced the execution time from 24 seconds to 6 seconds. This was because all the street name locations along one street, collected from OSM, were included as detected landmarks.

One important change that was added in later tests (see Section 5.2) was snapping the vehicle onto the closest road each time the movement was updated, which ensured that the vehicle was moving only on road sections. Interestingly, snapping the vehicle

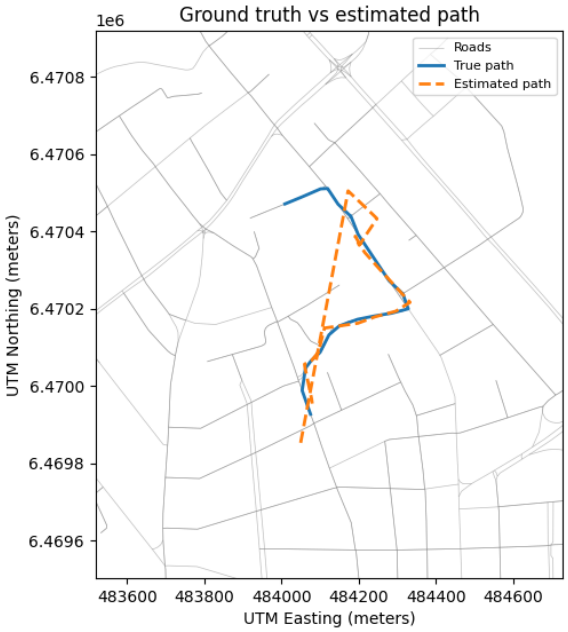


Figure 16. Comparison of ground truth (blue line) with estimated path (orange dashed line). These results were generated using 2500 particles with average error of 59 meters. Estimated locations were not snapped onto the road network, they were the average particle locations at given iteration step. As visible the estimated location starts from far away from ground truth location but is able to get closer after each iteration.

onto the closest road section did not improve localization accuracy. Furthermore, adding context to the closest road segment by including information about the previously chosen road did not improve accuracy.

In conclusion, including all the introduced modifications improved localization accuracy and reduced execution time. Filtering the landmarks based on distance to the estimated vehicle's location had the biggest effect on reducing the execution time. Experiments with only filtering landmarks based on vehicle orientation and snapping the vehicle onto the closest road section did not reduce execution time. For the latter, the execution time even increased.

5.5.2 Discussion on experiments with different amount of particles

Couple of experiments introduced in Section 5.4 were conducted based on the quantity of particles, which showed that the best results were obtained by initializing 3000 particles

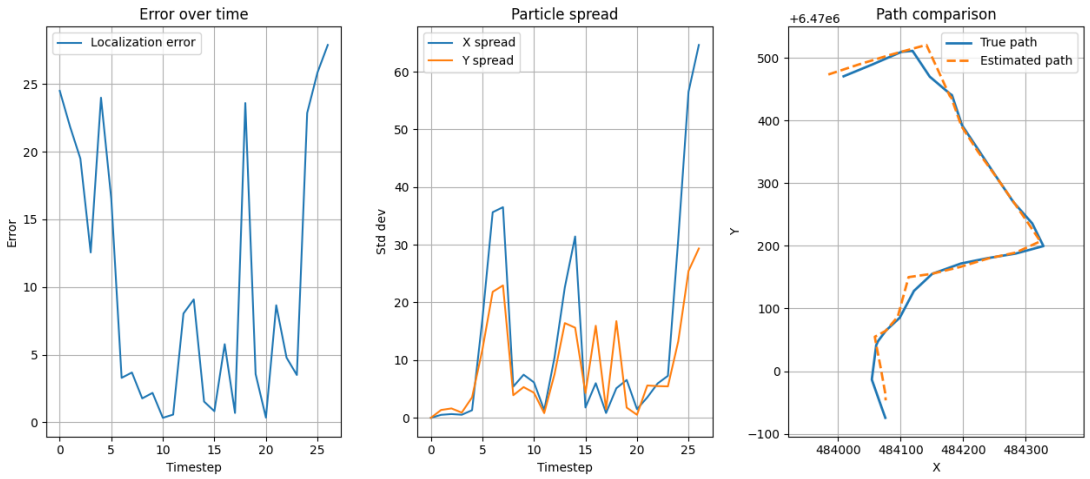


Figure 17. Error over time, particle spread and comparison of ground truth (blue line) with estimated path (orange dashed line). These results were generated using 3000 particles with average error of 10.28 meters.

Particles count	Average error (meters)	Std of error	Execution time (seconds)	Std of time
500	329.710	209.5	6.353	0.2
1000	225.837	186.4	11.642	1.2
1250	243.129	184.2	14.146	0.4
1500	276.632	195.5	16.847	1.3
2000	226.789	178.3	21.604	0.6
2250	265.247	178.4	26.243	3.5
2500	**183.560**	**184.1**	**27.391**	**0.8**
2750	153.417	152.0	32.608	7.6
3000	**157.489**	**144.4**	**56.926**	**6.2**
3250	210.018	160.9	59.367	2.0
3500	**152.342**	**133.1**	**51.476**	**11.1**
4000	263.475	205.8	64.543	11.3

Table 1. Comparison of different particle counts. Best results are achieved using 3500 particles which has the smallest standard deviation of error.

Experiments	Average error (meters)	Execution time (seconds)
Without filtering	361.96	24.08
Only front landmarks	370.14	20.22
Distance filtering	295.44	6.11
Snapping onto road	397.57	28.06
All modifications	**275.21**	**6.77**

Table 2. Comparison of different experiments using 2000 particles. Best results are achieved using all of the modifications.

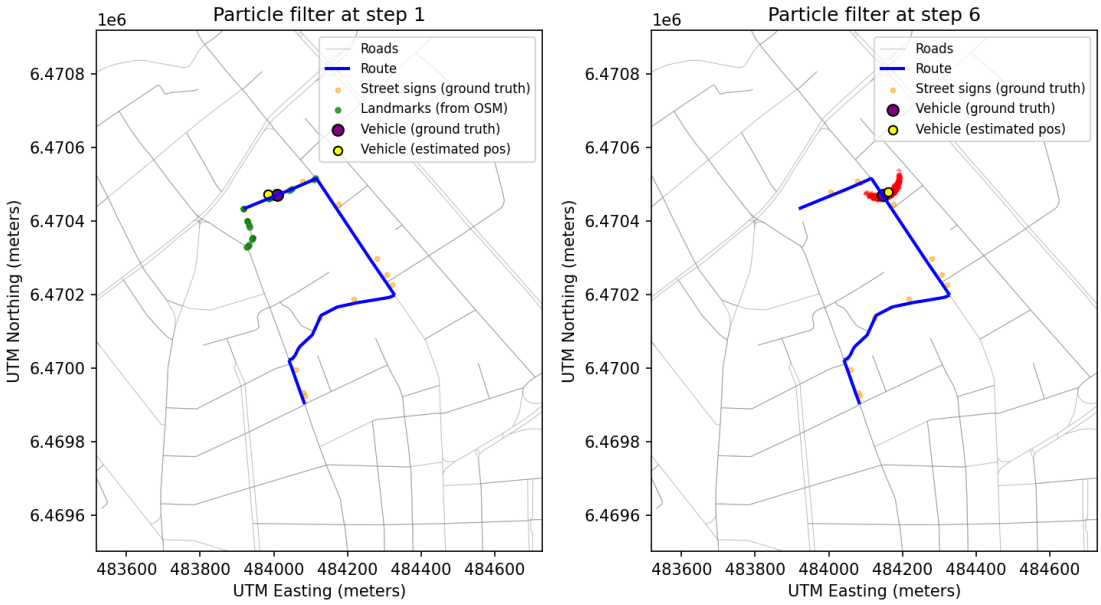
having the lowest error of 10.28 meters (during one experiment) and reasonably short execution time (see Figure 18). More details are visible in GIF[11]. The closest best result was achieved using 2500 particles that gave an average error of 59 meters (during one experiment).

Interestingly, while having the lowest average error and relatively similar execution time (compared to 3000), using 3500 particles gave the worst outcome, resulting in an average error of more than 200 meters. When inspecting the Figure 19 it is evident that the particles have diverged further away from the vehicle, therefore the estimated location is off by more than 200 meters. More information is visible in GIF[12].

Of course, it is important to note that it is more accurate to consider the average error of multiple experiments, which increases the error rates to 183 meters (for 2500 particles) and 157 meters (for 3000 particles), respectively. An important detail to point out is that, while inspecting Table 1, one can notice relatively large standard deviation values for error rates. This might be due to the nature of the particle filter, the limited number of test runs, and variability in initial particle placement. Although increasing test runs up to 100 did not significantly reduce the standard deviation.

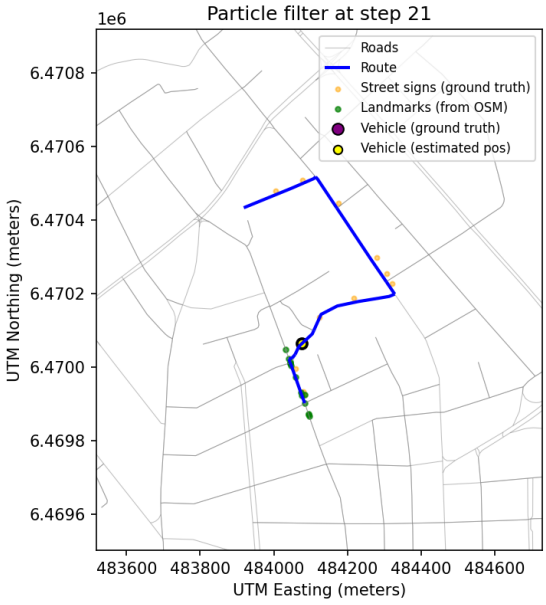
The main takeaway from these experiments is that the bigger the amount of particles, the longer the execution time. Moreover, improvement in accuracy is not that significant or cannot compromise the high execution time. The best results were achieved using 3000 particles, snapping the vehicle onto the closest road section and filtering out landmarks based on vehicle's orientation and the distance from the vehicle, resulting in 10 meter error during the best experiment run and an average error of 157 meters (over 20 experiments).

[11] https://tinyurl.com/ynn8t36t
[12] https://tinyurl.com/y5xbjbn5

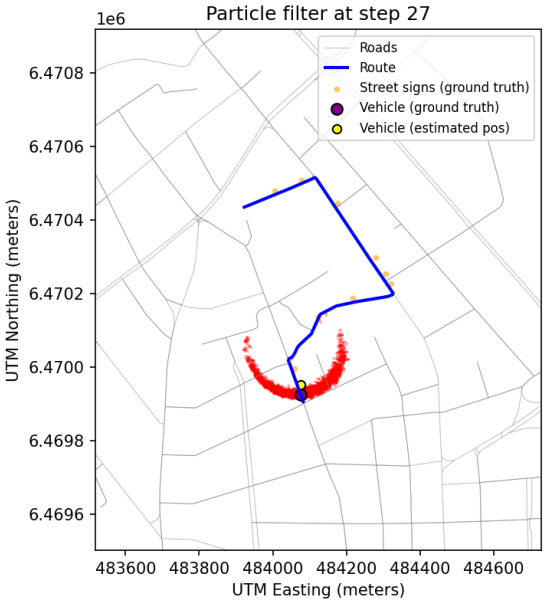


(a) Particle filter step 1

(b) Particle filter step 4



(c) Particle filter step 12



(d) Particle filter step 27

Figure 18. Particle filter steps with 3000 initialized particles.

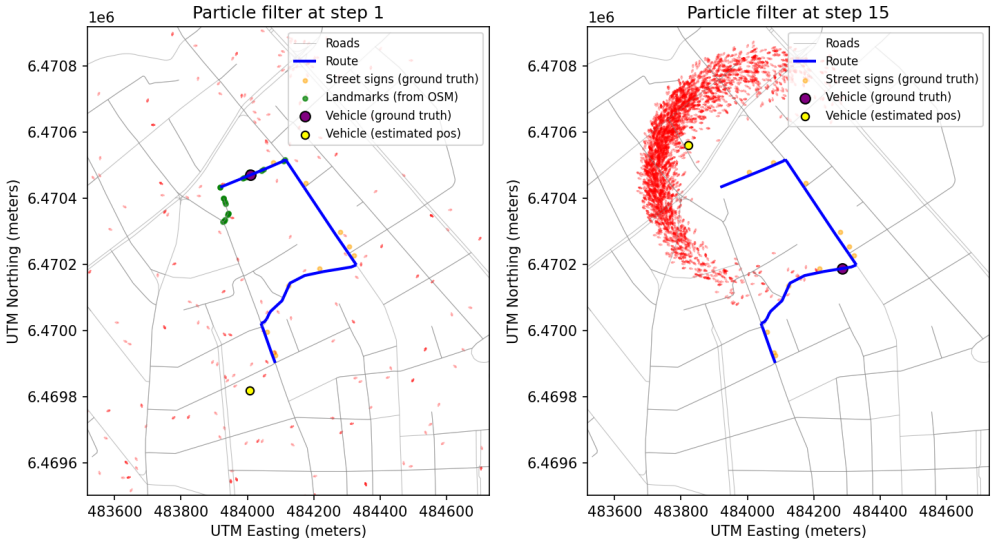


Figure 19. The first and 15th step of the particle filter with 3500 initialized particles. It is visible that the particles have diverged away from the vehicle increasing error rate. It is also visible that some particles have changed direction moving towards the vehicle's location.

5.5.3 Discussion on experiments with corrupted data

Additional testing included changing one or more landmarks with false information (Kalevi street was changed to Turu street, both located in Tartu) and using landmarks outside the map boundaries (two street name signs were out of the map). The main observations from these experiments were that introducing one or more false landmarks into the system did not affect particle filter that much because the additional filtering ruled out farther away landmarks. With out-of-bounds data, the vehicle was still snapped onto the closest roads, so it could not move outside of the map.

6 Conclusions and future work

6.1 Conclusions

Autonomous vehicles rely heavily on accurate self-localization to move safely and efficiently within their environment. Although satellite-based systems such as GNSS (e.g., GPS) can provide such vehicles with localization information, GNSS information might not always be available due to possible GNSS sensor failure or signal interference. One robust technique for vehicles to localize is using particle filters, given a map of the environment.

In this thesis a framework for robust urban-level localization was presented using vision-based detection and a particle filter. Despite the limitations of the given framework, it was possible to locate an autonomous vehicle without any information of the GNSS position with an error of 10 meters. Multiple experiments were conducted, resulting in the best choice for the amount of particles which was 3000 with an average error of 157 meters. Further filtering helped improve localization accuracy even more and gave promising results.

This framework, which can be found on GitHub[13] provides a good starting point for any future improvements and experiments on the problem of GNSS-free localization in autonomous vehicles.

6.2 Future work

In Section 4.2 the proposed framework was missing the street sign detection model. One possible future improvement that due to the time constraints was left out would be to implement the vision model for detecting street name signs and bus stops from street view images. In addition, including actual motion data in a form of ROS bag files would make the framework more reliable and accurate for real-world situations.

In Section 4.1 some basic assumptions were defined, which could be further improved. For example, the given framework could be generalized to other landmark types, such as traffic signs or traffic lights.

One possible future avenue would be the integration of the framework into the Autoware Mini system [14].

[13] GitHub repository, https://github.com/helenasokk/particle_filter.git
[14] Autoware Mini, https://adl.cs.ut.ee/lab/software

References

[ADR00] Kevin Murphy Arnaud Doucet, Nando de Freitas and Stuart Russen. Rao-blackwellised particle filtering for dynamic bayesian networks, 2000.

[Afi14] Deambrogio L. Salós D. Escher A.-C. Macabiau C. et al.. Afia, A. B. Review and classification of vision-based localisation techniques in unknown environments. *IET Radar Sonar and Navigation*, 8(9):1059–1072, 2014.

[Aka22] Naoki Akai. Reliable monte carlo localization for mobile robots, 2022.

[Ala23] Mansoor Alam. Localization in robotics for mobile robots. `https://medium.com/%40mansooralam129047/localization-in-robotics-for-mobile-robotsec3ad31f99d4`, 2023. accessed 14-01-2025.

[BGU16] Marcus A. Brubaker, Andreas Geiger, and Raquel Urtasun. Map-based probabilistic visual self-localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):652–665, 2016.

[BS16] Cattaneo D. Fontana S. Ballardini, A. L. and D. G. Sorrenti. Leveraging the osm building data to enhance the localization of an urban vehicle. *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, page 622–628, 2016.

[BWPU20] Ioan Andrei Bârsan, Shenlong Wang, Andrei Pokrovsky, and Raquel Urtasun. Learning to localize using a lidar intensity map, 2020.

[CB09] Mousset S. Nashashibi F. Challita, G. and A. Bensrhair. Particle filters for accurate localization of communicant vehicles using gps and vision systems. *IEEE/ACS International Conference on Computer Systems and Applications*, pages 238–242, 2009.

[CE24] Karim El Moutaouakil Vasile Palade Ali Yahyaouy-Uche Onyekpe Charroud, Anas and Eyo U. Eyo. Localization and mapping for self-driving vehicles: A survey, 2024.

[FL13] van der Zander B. Floros, G. and B. Leibe. Openstreetslam: Global vehicle localization using openstreetmaps. *IEEE International Conference on Robotics and Automation*, pages 1054–1059, 2013.

[HW14] Creusen I. M. Hazelhoff, L. and P. H. With. Exploiting street-level panoramic images for large-scale automated surveying of traffic signs. *Machine Vision and Applications*, 25:1893–1911, 2014.

[Int25] SAE International. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles, 2025. https://www.sae.org/binaries/content/assets/cm/content/blog/sae-j3016-visual-chart_5.3.21.pdf; accessed 03-05-2025.

[Kap96] E. D. Kaplan. *Understanding GPS: principles and characteristics*. Artech House, 1996.

[KM23] Debasis Kumar and Naveed Muhammad. A survey on localization for autonomous vehicles. *IEEE Access*, 11:115865–115883, 2023.

[Kor24] Kirsten Korosec. Zoox robotaxis are finally rolling out, waymo snags more cash, and aurora delays its self-driving truck launch, 2024. https://techcrunch.com/2024/10/31/zoox-robotaxis-are-finally-rolling-out-waymo-snags-more-cash-and-aurora-delays-its-self-driving-truck-launch/, accessed 08-05-2025.

[KPP+21] Bimsara Kanchana, Rojith Peiris, Damitha Perera, Dulani Jayasinghe, and Dharshana Kasthurirathna. Computer vision for autonomous driving. pages 175–180, 12 2021.

[KT01] Shin Kato and Sadayuki Tsugawa. Cooperative driving of autonomous vehicles based on localization, inter-vehicle communications and vision systems. *JSAE Review*, 22(4):503–509, 2001.

[LBAN19] Xing Liu, Tarig Ballal, and Tareq Al-Naffouri. Gnss-based localization for autonomous vehicles: Prospects and challenges. 09 2019.

[LCK+24] Youqi Liao, Xieyuanli Chen, Shuhao Kang, Jianping Li, Zhen Dong, Hongchao Fan, and Bisheng Yang. Osmloc: Single image-based visual localization in openstreetmap with geometric and semantic guidances, 2024.

[LCZT23] Lizhe Liu, Xiaohao Chen, Siyu Zhu, and Ping Tan. Condlanenet: a top-to-down lane detection framework based on conditional convolution, 2023.

[LKA+22] Johann Laconte, Abderrahim Kasmi, Romuald Aufrère, Maxime Vaidis, and Roland Chapuis. A survey of localization methods for autonomous vehicles in highway scenarios. *Sensors*, 22(1):247, 2022.

[LSY+22] Zhewei Liu, Wenzhong Shi, Yue Yu, Pengfei Chen, and Bi Yu Chen. A lstm-based approach for modelling the movement uncertainty of indoor trajectories with mobile sensing data. *International Journal of Applied Earth Observation and Geoinformation*, 108:102758, 2022.

[MCH11] Isaac Miller, Mark Campbell, and Dan Huttenlocher. Map-aided localization in sparse global positioning system environments using vision and particle filtering. *J. Field Robotics*, 28:619–643, 09 2011.

[NSHP21] Kana Nagai, Matthew Spenko, Ron Henderson, and Boris Pervan. Evaluating ins/gnss availability for self-driving cars in urban environments, 2021.

[ODM+24] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

[PCM16] Edward Pepperell, Peter Corke, and Michael Milford. Routed roads: Probabilistic vision-based place recognition for changing conditions, split streets and varied viewpoints. *The International Journal of Robotics Research*, 35:1057 – 1179, 2016.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[Ron15] Fischer P. Brox T. Ronneberger, O. U-net: Convolutional networks for biomedical image segmentation. *Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 9351, 2015.

[SA23] Inc. SuperAnnotate AI. Computer vision challenges in autonomous vehicles: The future of ai, 2023. https://www.superannotate.com/blog/computer-vision-in-autonomous-vehicles, accessed 07-05-2025.

[SB23] DeTone D. Yang T. Y. Avetisyan-A. Straub J. Malisiewicz T. ... Sarlin, P. E. and V. Balntas. Orienternet: Visual localization in 2d public maps with neural matching. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21632–21642, 2023.

[SMM23] Ali Hosseini Salari, Hossein Mirzaeinejad, and Majid Fooladi Mahani. Tire normal force estimation using artificial neural networks and fuzzy classifiers: Experimental validation. *Applied Soft Computing*, 132:109835, 2023.

[TC24] Lian Z. Wang P. Wang M. Yue-Z. Tian, Y. and H. Chai. Application of a long short-term memory neural network algorithm fused with kalman filter in uwb indoor positioning. *Scientific Reports*, 14:1925, 2024.

[Tes25] Tesla. Tesla vision update: Replacing ultrasonic sensors with tesla vision, 2025. https://www.tesla.com/support/transitioning-tesla-vision, accessed 07-05-2025.

[TF05] Burgard W. Thrun, S. and D. Fox. Probabilistic robotics, 2005. https://docs.ufpr.br/ danielsantos/ProbabilisticRobotics.pdf; accessed 13-01-2025.

[UT25] Inc. Uber Technologies. Uber teams up with nvidia to accelerate autonomous mobility, 2025. https://investor.uber.com/news-events/news/press-release-details/2025/Uber-Teams-Up-with-NVIDIA-to-Accelerate-Autonomous-Mobility/default.aspx, accessed 08-05-2025.

[WBP17] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric, 2017.

[WFMB19] Ami Woo, Baris Fidan, W.W. Melek, and R. Buehrer. Localization for autonomous driving. *Handbook of Position Location: Theory, Practice, and Advances*, (2):1051–1087, 01 2019.

[Wik25] OpenStreetMap Wiki. About openstreetmap — openstreetmap wiki,, 2025. https://wiki.openstreetmap.org/w/index.php?title=About_OpenStreetMapoldid=2818870; accessed 12-01-2025.

[WU19] Bârsan I. A. Wang S. Martinez J. Wei, X. and R. Urtasun. Learning to localize through compressed binary maps. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10308–10316, 2019.

[WZL+24] Hang Wu, Zhenghao Zhang, Siyuan Lin, Xiangru Mu, Qiang Zhao, Ming Yang, and Tong Qin. Maplocnet: Coarse-to-fine feature registration for visual re-localization in navigation maps, 2024.

[YKH+24] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data, 2024.

[YLCT19] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies, 06 2019.

[YS19] Vysotska O. Yan, F. and C. Stachniss. Global localization on openstreetmap using 4-bit semantic descriptors. *2019 European Conference on Mobile Robots (ECMR)*, pages 1–7, 2019.

[ZCSA+21] Mengjie Zhou, Xieyuanli Chen, Obed N. Samano Abonce, Cyrill Stachniss, and Andrew Calway. Efficient localisation using images and openstreetmaps. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021. IEEE/RSJ International Conference on Intelligent Robots and Systems ; Conference date: 27-09-2021 Through 01-10-2021.

[ZDP+21] Chongsheng Zhang, Weiping Ding, Guowen Peng, Feifei Fu, and Wei Wang. Street view text recognition with deep learning for urban scene understanding in intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4727–4743, 2021.

[ZFZ+21] Tu Zheng, Hao Fang, Yi Zhang, Wenjian Tang, Zheng Yang, Haifeng Liu, and Deng Cai. Resa: Recurrent feature-shift aggregator for lane detection, 2021.

[ZL21] Fan H. Zhang, C. and W. Li. Automated detecting and placing road objects from street-level images. *Computational Urban Science*, 1(18), 2021.

[ZTD+22] Chongsheng Zhang, Yuefeng Tao, Kai Du, Weiping Ding, Bin Wang, Ji Liu, and Wei Wang. Character-level street view text spotting based on deep multisegmentation network for smarter autonomous driving. *IEEE Transactions on Artificial Intelligence*, 3(2):297–308, 2022.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Helena Sokk**,

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

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Vision-based localization on city scale using Open Street Map,

   supervised by Naveed Muhammad and Dmytro Zabolotnii.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Helena Sokk
15/05/2025