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

Erik Mägi

Georeferenced Visual SLAM

Master's Thesis (30 ECTS)

Supervisor(s): Tambet Matiisen, MSc
Edgar Sepp, MSc

Tartu 2023

Georeferenced Visual SLAM

Abstract:
This thesis presents a complementary localization solution for taxis and ride-hailing operators in situations where GNSS is unavailable or unreliable. The proposed method leverages monocular visual SLAM techniques, specifically the ORB-SLAM 3 library, to create a map of the environment and localize within it. The system uses a car-mounted camera for image capture and an advanced GNSS receiver to record accurate ground truth. This data is then used as input for training a deep learning model to transform SLAM coordinates into georeferenced coordinates. The thesis explores different approaches to solving the coordinate transformation problem, including linear transformation, machine learning regression algorithms, and deep learning with neural networks. Results show that the deep learning based approach provides the best localization accuracy, surpassing that of modern smartphone GNSS. The study contributes a practical solution for real-time localization for ride-hailing operators when GNSS is compromised, with the potential for future implementation using smartphone cameras.

Keywords:
Visual SLAM, ORB-SLAM 3, georeferencing, neural networks.

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

Georefereeritud visuaalne SLAM

Lühikokkuvõte:
Käesolev magistritöö esitab täiendava asukoha määramise lahenduse taksodele ja sõidujagamise teenusepakkujatele olukordadeks, kus GNSS signaal puudub või on ebausaldusväärne. Pakutav meetod kasutab monokulaarse visuaalse SLAM-i meetodeid, täpsemalt ORB-SLAM 3 teeki, et kaardistada keskkond ja selles siis asukoht määrata. Süsteem kasutab pildi saamiseks auto külge kinnitatud kaamerat ja kõrge täpsusega GNSS vastuvõtjat, et salvestada tegelik liikumistrajektoor. Nende andmetega treenitakse sügavõpe mudel, mis teisendab SLAM-koordinaadid reaalmaailma koordinaatidesse. See töö uurib erinevaid lähenemisi koordinaatide teisendamiseks, sealhulgas lineaarset teisendust, masinõppe regressioonialgoritme ja sügavõpet närvivõrkudega. Tulemused näitavad, et sügavõppe meetodil põhinev lahendus pakub parimat asukoha määramise täpsust, ületades kaasaegse nutitelefoni GNSS-i täpsuse. Uuring annab praktilise lahenduse reaalajas asukoha määramiseks sõidujagamise teenusepakkujatele, kui GNSS ei ole kasutatav, ning loob võimaluse nutitelefoni kaamera põhise lokaliseerimissüsteemi juurutamiseks.

Võtmesõnad:
Visuaalne SLAM, ORB-SLAM 3, georefereerimine, närvivõrgud.

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

Contents

1 Introduction

Location-based services and the ability to quickly and reliably pinpoint our position have become ubiquitous features of modern life. Most often, we use smartphones to interact with these services and rely on the built-in GNSS receivers for accurate localization. While this may be sufficient for getting directions from point A to point B or geotagging photos, the localization accuracy offered by these devices can become problematic for use cases that require sub-meter precision. One such use case is ride-hailing, more specifically, matching and guiding drivers to passengers.

The main challenge with GNSS positioning in general is its heavy dependence on the operating environment of the receiver, including factors such as the number of visible satellites, cloud density, atmospheric conditions, proximity to high buildings, and dense foliage. This is further exacerbated by the fact that the GNSS receivers in smartphones are relatively weak due to constraints imposed by small, battery-operated handheld devices. Although there are several tricks and techniques to work around these limitations, such as network-based positioning (both cellular and Wi-Fi) and the use of inertial sensors to compensate for noisy signals, the localization error still remains above 3 meters for even the best smartphones [4].

Ride-hailing has become an extremely popular service, particularly in urban areas where owning a personal car is either prohibitively expensive or impractical due to heavy traffic. The process is simple: using an app, a person requests to be taken from point A to point B. The service then matches the passenger with a driver, generates directions for the driver to the passenger, and provides an ETA for the passenger. Drivers use smartphones for accepting requests and navigation, so directions and ETA are generated from the positional data provided by the smartphone. However, a localization error of 3 meters could result in the driver being positioned in the opposite lane of traffic on the map, leading to incorrect directions and ETA, unclear instructions for the driver, and a wrong ETA for the passenger. This translates to lowered customer satisfaction and, potentially, lost revenue for the service provider.

In this work, we explore improving localization accuracy for on-road vehicles by leveraging the smartphone camera and other sensors. We set the following constraints for the solution: it must be easy to integrate into the existing infrastructure, preferably requiring only a smartphone, which each driver already has in their car; it must work with monocular cameras; and the solution should operate in real-time or near real-time. After trying multiple approaches, we eventually arrive at a solution based on the open-source ORB-SLAM 3 [1] library for mapping and localization, with an additional post-processing step using neural networks to correct scaling issues inherent to monocular SLAM and translate the obtained localization result from ORB-SLAM 3 to real-world coordinates.

The proposed solution is not yet a production-ready system but rather a proof of concept that demonstrates the possibility and viability of our approach for improving

localization accuracy. This thesis is organized into six sections, which we will briefly describe here.

- *Background and Related Work* covers other algorithms and systems that were considered.
- *Problem Statement and Objectives* provides a deeper look into the problem at hand and motivation.
- *Proposed Solution* discusses the process of reaching our current solution, including alternative routes that we explored.
- *Evaluation and Results* describes the experiments and summarizes the results.
- *Discussion* discusses the results and names possible directions for future work.
- *Conclusion* provides a summary for the work presented in this thesis and lists released source code that was created in the scope of this work.

2 Background and Related Work

In this section, we provide a brief overview of the core concepts and techniques that underlie Structure from Motion (SfM) and Simultaneous Localization and Mapping (SLAM), which we considered as the foundation for our solution. Following this, we take a more in-depth look at the state of the art in visual localization and mapping.

2.1 3D reconstruction

3D reconstruction is a well studied problem in computer vision and robotics that has occupied researchers for over 30 years. The essensce of the problem can be succintly described with the following quote:

> *Given a set of photographs of an object or a scene, estimate the most likely 3D shape that explains those photographs, under the assumptions of known materials, viewpoints, and lighting conditions*. [7]

SfM and SLAM are methods that are used throughout this work. We will provide a high-level overview of both methods to offer additional context for the reader.

SfM SfM is a 3D reconstruction technique (Figure 1) that estimates the three-dimensional structure of a scene from a set of potentially unordered images taken from different viewpoints. The process involves extracting and matching features across multiple images, followed by estimating the camera poses and 3D points through bundle adjustment. SfM can work offline and is often used for applications such as photogrammetry, 3D modeling, and large-scale reconstruction (3D modeling of large areas).

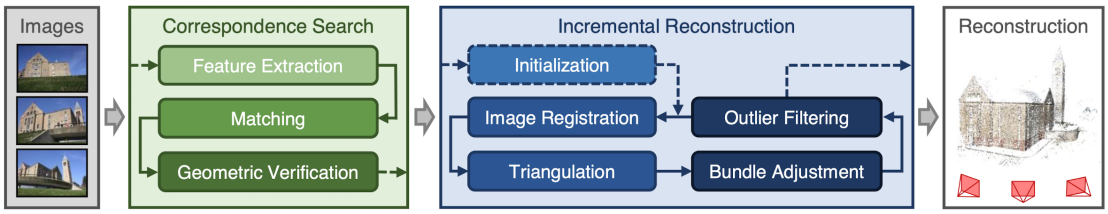


Figure 1. Incremental SfM pipeline [26]. While this figure depicts incremental reconstruction, the core of the method remains the same: unordered set of images is used to search feature correspondences, then camera poses and 3D points are estimated through bundle adjustment.

SLAM In this work, we focus on visual SLAM, which uses only visual data (images) to estimate the camera trajectory and map. For brevity we will refer to visual SLAM as SLAM throughout the rest of this work.

SLAM is another 3D reconstruction method (Figure 2) that estimates the camera trajectory and maps the environment simultaneously. Unlike SfM, SLAM is designed to work in real-time and is more suitable for robotics and autonomous navigation applications. Visual SLAM uses various algorithms, such as keypoint detection, matching, and tracking, along with loop closure and bundle adjustment to maintain a consistent map and estimate camera poses. Visual SLAM algorithms can be broadly categorized into two groups: feature-based and direct. Feature-based methods preprocess the frame, extract features, and build keypoints from correspondences between frames to estimate camera poses and 3D points. In contrast with the feature-based methods, direct methods use raw pixel intensities to minimize the photometric error between frames. Direct methods can yield more accurate results but are tightly coupled with the camera model and are computationaly more expensive.

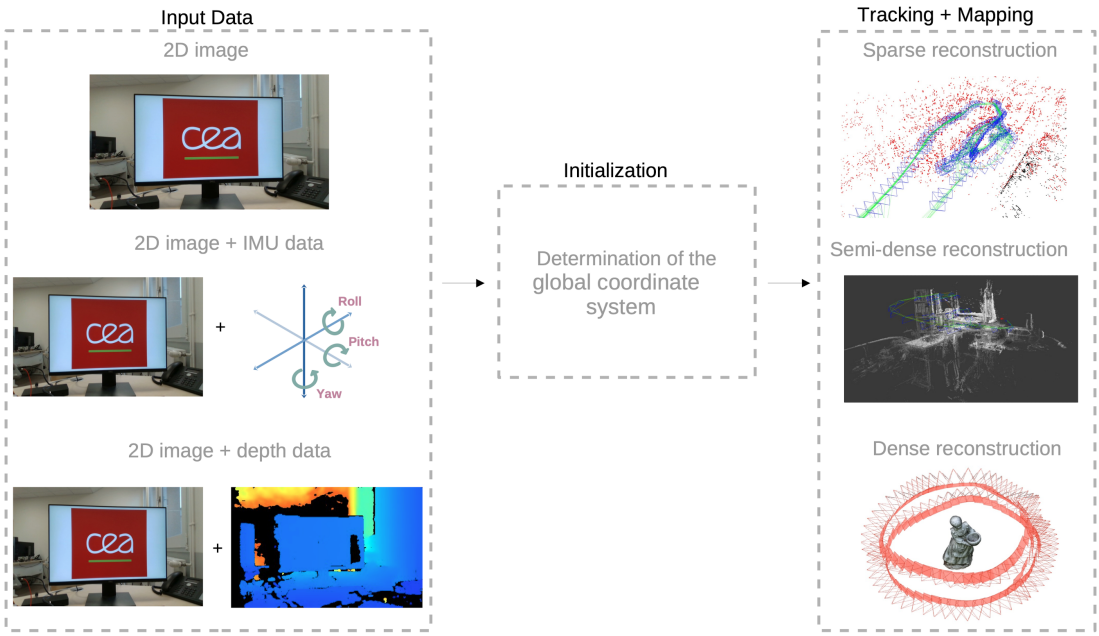


Figure 2. General components of a visual-based SLAM [18]. Inertial (IMU) and depth data (RBG-D) can be added to improve the accuracy of the system.

While both SfM and SLAM aim to reconstruct 3D geometry from a series of images, there are some key differences between the two methods:

- Real-time operation: SLAM is designed to work in real-time, making it suitable

for robotics and autonomous navigation applications. In contrast, SfM is typically used in offline processing, prioritizing 3D model accuracy and reconstruction quality.

- Mapping and localization: SLAM simultaneously estimates the camera trajectory and builds a map of the environment. SfM, on the other hand, is primarily concerned with generating a 3D reconstruction without explicitly focusing on localization.

2.2 Literature overview

2.2.1 Collaborative Augmented Reality on Smartphones via Life-long City-scale Maps

Platinsky et al. [25] present a highly accurate, large-scale, end-to-end system for mapping, map management, and localization with sub-meter accuracy. Their main use case involves shared augmented reality (AR) experiences for pedestrians, such as games and annotated points of interest.

The proposed system consists of two main parts: the mapping and localization pipelines (Figure 3). Mapping is a multi-step process:

1. Specialized vehicles equipped with multiple cameras and high-precision GNSS receivers capture image and location data.
2. This data is divided into smaller units, each covering an area with a radius of approximately 20 meters.
3. Each split is processed using Structure from Motion (SfM) to reconstruct the 3D geometry of the mapped area.
4. Eventually, all splits are merged into one continuous map.
5. Map changes are stored in a cloud service.

Since the map is stored in a remote service, direct image-based localization is not possible. Instead, localization is done using a visual-inertial odometry system running on each edge device, such as a smartphone. Each edge device tracks its own location and is able to transform it into a global coordinate frame, sending keyframes to the remote map server. Due to the precise base map obtained through SfM, the system achieves high localization accuracy with a mean error of 42 cm and a median error of 18 cm.

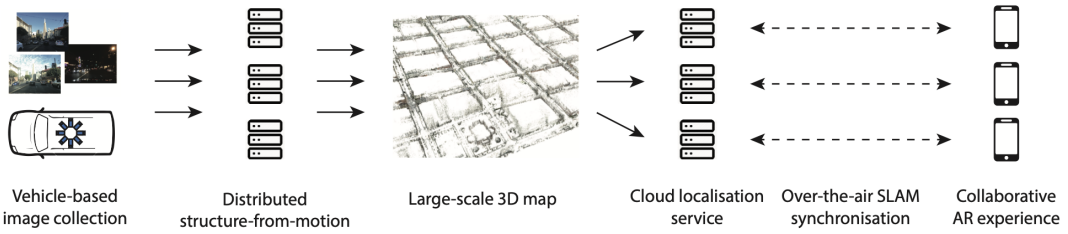


Figure 3. High-level overview of the system. A large-scale 3D map is computed from a vehicle-based image collection and used to localise camera-equipped edge devices for AR experiences over a mobile network [25].

We explored SfM briefly in our work, which we describe in Section 4.2 (OpenSfM) but decided against creating an SfM-based system due to its offline nature and the need for specialized hardware to capture high quality data for reconstruction. Both of these factors contradict the requirements we set for our system, as discussed in Section 3.2 (Goal).

2.2.2 Uncertainty-aware Vision-based Metric Cross-view Geolocalization

Fervers et al. [6] propose a novel method for vision-based metric cross-view geolocalization (CVGL) that matches camera images captured from a ground-based vehicle with an aerial image to determine the vehicle's position in the real world (Figure 4).

This approach first constructs a bird's eye view (BEV) representation of the vehicle's on-ground view using a cross-attention mechanism. The BEV is then matched with the aerial image to determine the vehicle's position. The proposed method achieves a mean Absolute Pose Error (APE) of 0.78 m on the KITTI-360 dataset [17].

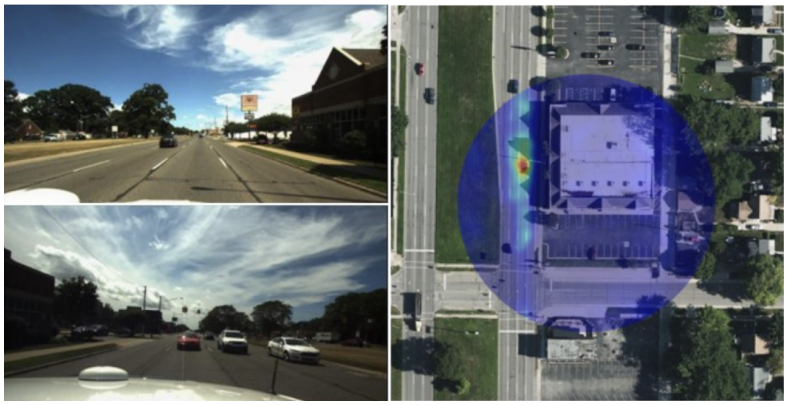


Figure 4. Probability distribution for the vehicle matching its camera input with an aerial image [6].

While this is certainly a novel approach, it does require additional hardware to be installed on the vehicle to get a 360-degree view of the surroundings for best results. Availability and currentness of aerial imagery is different in different regions is another concern that limits the practicality of this approach for our use case. In addition, using aerial imagery at a scale could become costly due to licensing fees.

2.2.3 Street View Motion-from-Structure-from-Motion

Klingner et al.'s work [15] concentrates on creating large-scale point clouds from Google Street View data using SfM techniques and enhancing the point cloud accuracy by incorporating inertial data (Figure 5).

Google Street View images are captured using vehicles outfitted with various sensors (inertial, speed, GPS) in addition to cameras. This sensor data is fused to establish an initial trajectory. The authors leverage two properties of this initial pose to improve SfM reconstruction accuracy:

- The absolute pose error is up to 10 m due to multipath[1] GPS issues in dense urban areas.
- However, the relative pose along the vehicle path is very accurate (sub-centimeter) because it is constructed using calibrated IMU data over short time scales.

[1]The phenomenon in which GPS signals reflect off various surfaces before reaching the receiver, resulting in multiple paths for the signal to arrive at the receiver. This causes delays in signal reception and inaccuracies in the calculated position.

Figure 5. A comparison of vehicle paths before (a) and after (b) correction [15].

While this approach is not directly applicable to our work because it relies on Google Street View data, which is not freely available, it does suggest that an IMU-tracked path can be very accurate over short time scales. We can leverage this insight, and we discuss our experiments in Section 4.6.1 (IMU Integration). Google provides the ARCore Geospatial API[2], which is built on parts of this work. However, using this API on the scale of a ride-hailing company would incur significant licensing costs, which could make it an unviable approach for the task at hand.

2.2.4 ORB-SLAM 3

ORB-SLAM 3 [1] is a state-of-the-art feature-based SLAM system that supports visual, visual-inertial and multi-map SLAM with monocular, stereo and RGB-D cameras. As the latest addition to the ORB-SLAM family of SLAM systems [22, 20], it builds upon ORB-SLAM 2 [22] and ORB-SLAM VI [21].

[2] `https://developers.google.com/ar/develop/geospatial`

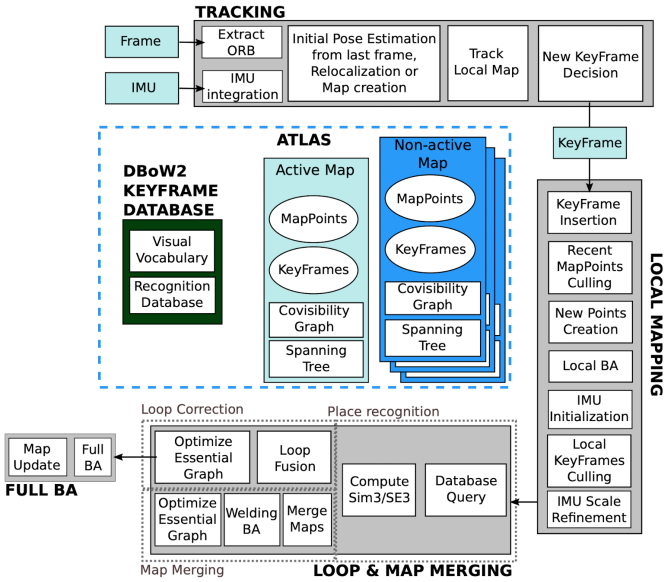


Figure 6. Main components of ORB-SLAM 3 [1].

ORB-SLAM 3 supports multiple camera configurations: monocular, stereo and RGB-D; as well as IMU integration for improved accuracy. It also supports loop closure that is especially helpful for monocular-only case. ORB-SLAM 3 is composed of five main components: input abstraction, tracking, local mapping, loop and map merging, and an internal map manager, the Atlas [5] (Figure 6). We will provide a brief overview of each component below:

- **Input abstraction** detaches the system from the specific input sensor type. Frames undergo pre-processing, ORB features are extracted, and all system operations are based on these features.
- **Tracking** calculates the current pose relative to the active map.
- **Local mapping** incorporates keyframes and map points into the local map, eliminates redundant ones, and performs local bundle adjustment (BA) within a local window of keyframes close to the current keyframe.
- **Loop and map merging** identifies common regions between the active map and other maps stored in Atlas. If the common area is part of the active map, loop correction occurs; if it belongs to a separate map, the two maps merge.
- **Atlas** is a multi-map representation consisting of a set of disconnected maps. Among these maps, one is designated as the active map for tracking and localization, while the others are considered non-active.

Our solution is partly based on ORB-SLAM 3 as it is a versatile system that can be used in a variety of scenarios. We provide a more comprehensive reasoning into this in Section 4.2 (Evolution of the Solution).

2.2.5 GPS-SLAM

Kiss-Illés et al. [14] propose a modified ORB-SLAM system that uses both IMU and GPS data to improve tracking performance. Their use case concerns mapping large-scale outdoor environments with unmanned aerial vehicles (UAVs) as the primary platform. Since, in their case, the UAVs record video only at frame rates in the range between 0.5 and 4 fps, they note that ORB-SLAM has problems with tracking due to poor continuity. They propose a modified ORB-SLAM system that uses IMU and GPS data to improve tracking performance. Unfortunately, they do not publish any results on the performance of their system in terms of trajectory accuracy. They only evaluate the quality of the generated maps.

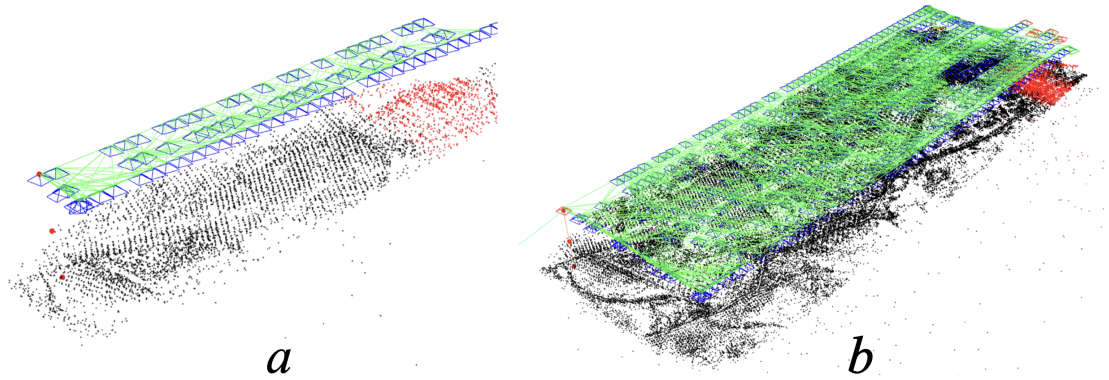


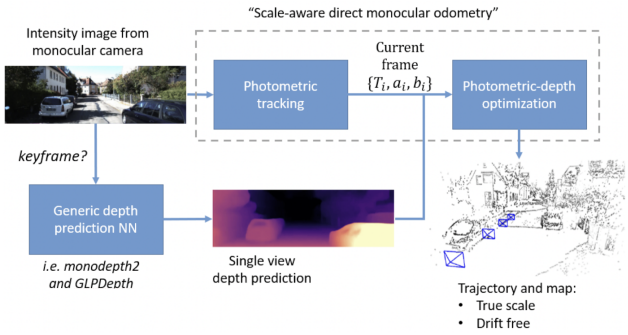
Figure 7. Differences between the original ORB-SLAM (a) and the modified one by Kiss-Illés et al. [14].

This work appears promising, but since it did not publish any results on trajectory accuracy and depends on IMU data, which we were unable to integrate into our system, we did not pursue this approach further. However, it remains an option for future work.

2.2.6 Scale-aware Direct Monocular Odometry

In this work, Campos et al. [2] propose a scale-aware direct monocular odometry system that uses a Convolutional Neural Network (CNN) to estimate pixel-wise depth in images. This estimation allows them to determine the true scale of the map, similar to stereo and visual-inertial systems. Furthermore, this system can completely eliminate the most significant issue in pure monocular systems: scale drift. They compare their method

against ORB-SLAM 3 with loop closure disabled on the KITTI[8] dataset and achieve significant improvements in the monocular case (Figure 8).



	Pure monocular		Monocular learning-based			Stereo
	ORB-SLAM (No LC)‡	DSO*	DF-VO [10]	Ours *GLP* [15]	Ours *monodepth2* [10]	ORB-SLAM (No LC)‡
00	77.19	113.18	11.34	10.52	**7.10**	3.99
01	110.12	-	484.86	**17.12**	25.38	1.38
02	34.32	116.81	21.16	31.16	**14.12**	8.82
03	0.90	1.39	2.04	3.41	**1.71**	0.25
04	0.72	0.42	0.86	0.31	**0.29**	0.22
05	36.29	47.46	**3.63**	5.61	7.51	2.18
06	52.61	55.62	**2.53**	6.23	4.05	1.81
07	17.04	16.72	**1.72**	3.45	2.41	1.43
08	56.42	111.08	**5.66**	9.30	10.43	3.22
09	55.74	52.23	10.88	11.32	**9.15**	3.26
10	8.44	11.09	3.72	4.36	**3.40**	0.88
Avg.†	33.97	52.6	6.35	8.57	**6,02**	2.60

*: Results for DSO are extracted from [10]
†: KITTI01 is not used for average to ease comparison between methods
‡: For ORB-SLAM we use the latest implementation from [4]

Figure 8. Overview of the system and performance comparison to ORB-SLAM 3 monocular. [2]. GLP [13] and monodepth2 [9] are depth estimation models.

Due to time constraints, we were unable to investigate this approach further. However, it remains a potential avenue for future work. It would be of interest to evaluate the performance of this approach when integrated into a feature-based system, such as ORB-SLAM 3.

3 Problem Statement and Objectives

Accurate positional data is essential for the operation of a ride-hailing service, or any type of service that relies on location-based functionality and optimal route computation. In the context of ride-hailing, there are two high-cost scenarios that are directly impacted by poor or unreliable positional data: optimal route planning and ETA calculation. In this section, we will explain the importance of these scenarios and outline objectives to address the challenges they present.

3.1 Motivation

Modern smartphones are versatile devices that combine substantial computational power and a wide array of sensors within a compact, portable form factor. This versatility allows them to be employed across various professional applications such as sales, surveying and mapping, emergency services, and transportation and logistics. Ride-hailing services use smartphones to localize both drivers and passengers.

We will provide a high-level overview of the main use cases for smartphones for both the driver and the passenger to establish a common understanding of the typical scenarios.

A passenger uses a customer version of the smartphone application to request a ride. When a passenger requests a ride, they are matched with a driver and presented with an ETA for when their ride will arrive at their location.

Each driver has a smartphone in their car that runs the driver version of the ride-hailing application, which tracks their location in real-time. The ride-hailing back-end service uses this information to match drivers with ride requests. When a driver accepts a ride request, the driver app calculates the optimal route to the passenger, sends the ETA to the back-end, which then forwards it to the passenger.

As noted earlier, the error of smartphone GNSS receivers ranges from around 3.5 meters in best-case scenarios to over 50 meters in worst-case scenarios [4]. In addition to environmental factors such as clouds, foliage, and tall buildings, accuracy also depends on phone hardware (receiver and chipset models) and software (drivers, applications). Due to these factors, the localization accuracy of smartphones is often insufficient for precisely localizing the driver, potentially positioning them on a neighboring road or at the wrong end of an intersection. Two examples illustrating such a scenario can be seen in Figure 9. This inaccuracy may lead to the calculation of incorrect or suboptimal routes and the delivery of incorrect ETAs to passengers.

Even though localization errors impact only approximately 1% of the rides[3], this issue still represents a considerable optimization opportunity for ride-hailing companies, which manage millions of rides daily [11]. In addition to the direct negative business

[3]Bolt, personal communication

impact for the operators, such as dissatisfied and potentially lost customers, increased operational costs due to higher fuel consumption and vehicle wear, these problems also have an environmental aspect.

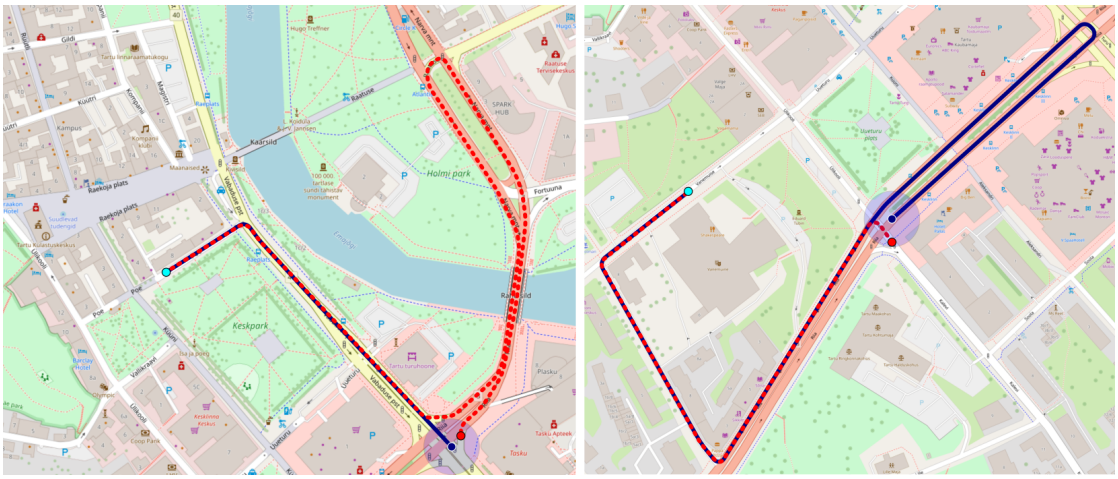


Figure 9. Illustration of miscalculated routes caused by poor localization accuracy. (Left) The calculated route based on the GPS position (blue) is considerably shorter than the route based on the actual position (red), leading to an underestimated ETA. (Right) The route derived from the GPS position is significantly longer than the actual route, resulting in an overestimated ETA.

In summary, smartphone GNSS accuracy is occasionally insufficient for ride-hailing services, leading to suboptimal route planning and under- or overestimation of ETAs. At scale, such inaccuracies can negatively impact the business by affecting customer satisfaction, increasing operational costs, and potentially harming the environment.

3.2 Goal

The primary goal of this thesis is to explore the possibility of a camera-based system with minimal hardware and integration requirements that would improve the localization accuracy of ride-hailing drivers or serve as a reliable fallback option when GNSS is unavailable or inadequate. The system would localize based on visual features obtained from the camera input (Figure 10).

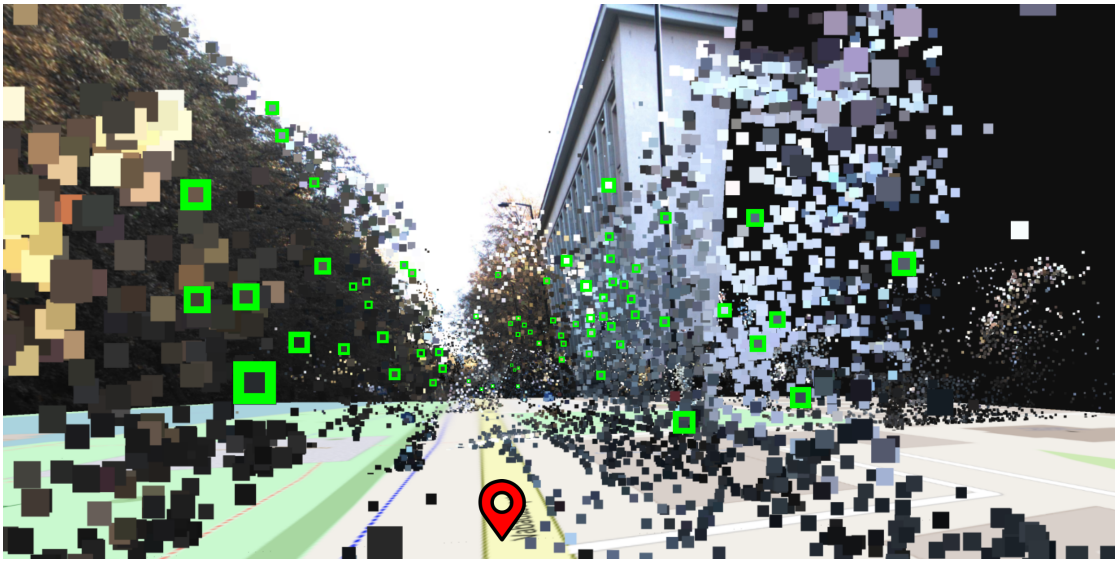


Figure 10. High level conceptual illustration of the goal of this thesis, a system that can localize itself using features read from camera input. The blocks are features extracted from a camera frame and each feature has coordinates attached to it that determine its position in the world. When doing localization using these features, the system is able to determine its position in the world on the map below.

To ensure the feasibility and practicality of the proposed solution, we have defined the following constraints that the system must satisfy:

1. It must be easily deployable to the existing fleet of vehicles. That is, it should not introduce any new hardware or other integration requirements. The solution should be entirely contained within the driver app.
2. The system should not introduce any operational overhead to the drivers and should be as seamless as possible.
3. The system must utilize the phone's main camera as the primary input source for visual data.
4. The system can leverage inertial sensors available on the phone to enhance localization accuracy.
5. Localization must happen in real-time or close to real-time.
6. It would be preferable if the solution would not require specialized vehicles or equipment for mapping the environment, and it could be done entirely by the existing fleet of vehicles.

4 Proposed Solution

In this section, we will introduce our proposed solution, outlining the development process, addressing its limitations, and discussing the challenges encountered during its implementation.

4.1 Overview

We propose a system that leverages a map created using a monocular visual SLAM system to improve localization capabilities for drivers, especially in situations where GNSS data is unreliable or of poor quality. Our system is comprised of two main components: the SLAM system, ORB-SLAM 3, and a Fully Connected Neural Network (FCNN) deep learning model for transforming localized position from SLAM coordinate space into real-world UTM (Universal Transverse Mercator) coordinate space, thereby providing georeferenced localization, Figure 11 gives a high level overview of the proposed system.

We base this solution on the hypothesis that given an accurate enough ground truth, it is possible to train a model that would provide considerably higher accuracy than smartphone GPS when localizing using SLAM features. In order to obtain a high-quality ground truth, we use the NovAtel PwrPak7D-E2, which offers centimeter-level precision through GNSS RTK. Camera input is captured using a Sekonix SF3324 RGB camera. These sensors are attached to our test vehicle[4] and the readings are recorded while participating in normal traffic. We expect the system to be used on roads, so we do not consider pedestrian localization in this thesis.

[4] https://adl.cs.ut.ee/lab/vehicle

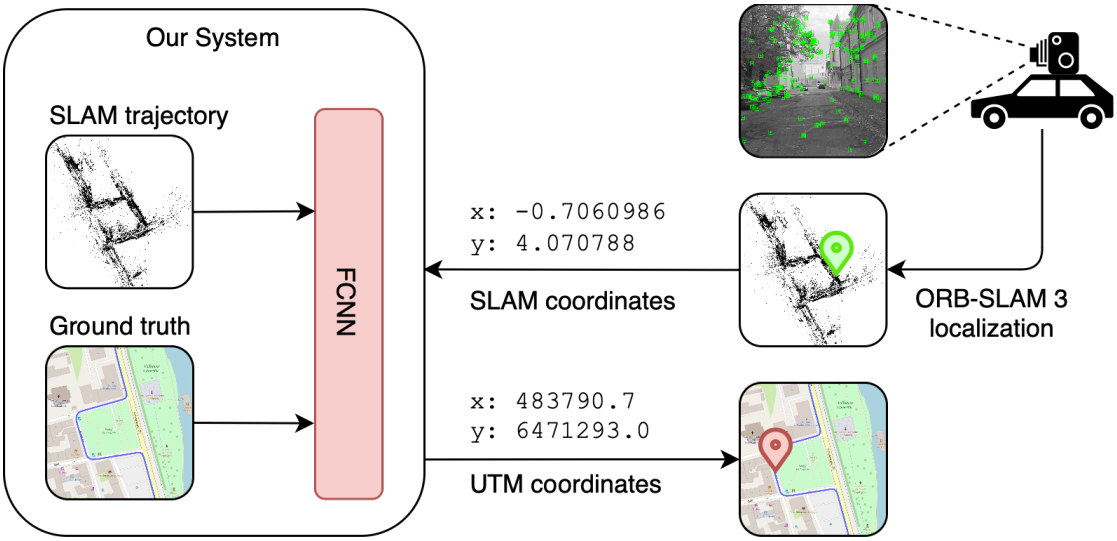


Figure 11. High level overview of the proposed system. The FCNN model is trained using the ground truth and SLAM data. The trained model is then used to transform SLAM localization coordinates into real-world coordinates.

It is important to note that the ground truth is provided in WGS84 (World Geodetic System 1984) coordinates: latitude, longitude, and altitude. Since the SLAM coordinate system is Cartesian and WGS84 is ellipsoidal, a simple translation, rotation, and scaling are not sufficient for transforming between the two systems. To address this issue, we convert the ground truth to the UTM (more precisely, UTM35[5]) Cartesian coordinate system, which also has the added benefit of representing x and y coordinate values in meters.

While our approach differs from other methods that try to improve SLAM accuracy by modifying the SLAM pipeline itself [14, 2], it meets the requirements we set for our use cases. By addressing the issue with an external component, our method aligns with the premise that the ride-hailing service's back-end server will be used to run localization queries. Although the mapping process requires additional high-precision GNSS hardware, not all vehicles need to be equipped with it, resulting in a minimal and unobtrusive change. Theoretically, any off-the-shelf SLAM solution could be used with this approach, assuming an alternative to Atlas is found for map storage. Furthermore, ORB-SLAM 3 has been demonstrated to work well on smartphones [28], making it suitable for scenarios requiring real-time or near-real-time localization.

It is important to note that, due to time constraints and various technical challenges, the current implementation serves as a proof of concept.

[5] (https://epsg.io/32635)

4.2 Evolution of the Solution

During our research, we evaluated and experimented with various methods. In this section, we will provide a brief overview of the two most promising candidates that we considered.

OpenSfM Motivated by the highly accurate results of the system by Platinsky et al. [25], we also considered a SfM based solution. We chose OpenSFM[6] as the basis.

OpenSfM (Open Source Structure from Motion) is an actively developed open-source library for performing SfM tasks. It is developed by Mapillary (now a subsidiary of Facebook), a street-level imagery platform that uses crowdsourced images to create detailed and up-to-date maps. In addition to tooling, such as OpenSfM, Mapillary also provides benchmark datasets such as CrowdDriven [12].

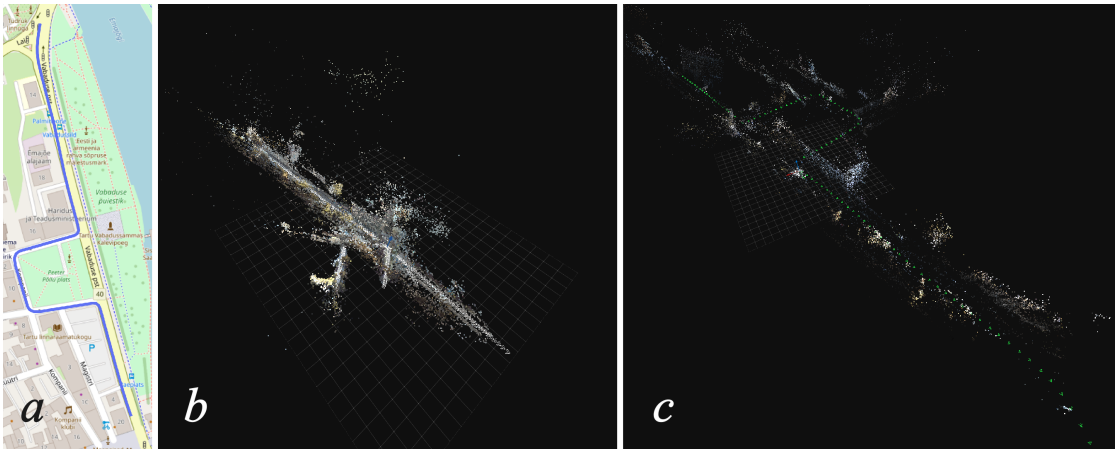


Figure 12. Reconstruction using OpenSfM. (a) Ground truth. (b) Reconstruction using image data only. (c) Reconstruction using images with embedded metadata containing GPS coordinates and time of creation. In panel (b), the reconstructed trajectory has multiple overlapping sections and does not resemble the ground truth. In panel (c), the correctly reconstructed trajectory can be observed, with the imaginary line through the green camera markers closely resembling the ground truth.

We initially assumed that OpenSfM would be able to correctly reconstruct the trajectory based on image data only since the images were ordered and consecutive images had many correspondences. However, this was not the case, as can be seen in Figure 12 (b); the system was able to correctly reconstruct sections of the trajectory but was unable to correctly merge them. Adding GPS location data and image creation timestamps via

[6] https://github.com/mapillary/OpenSfM

EXIF metadata significantly improved the results, and the entire trajectory was reconstructed correctly. While SfM offers great reconstruction accuracy, as can be seen in Figure 13, it is a slow process that needs to be performed offline. In addition to that, SfM is at its core a 3D geometry reconstruction algorithm and does not concern itself with localization. As such, also OpenSfM does not offer any built-in localization functionality.

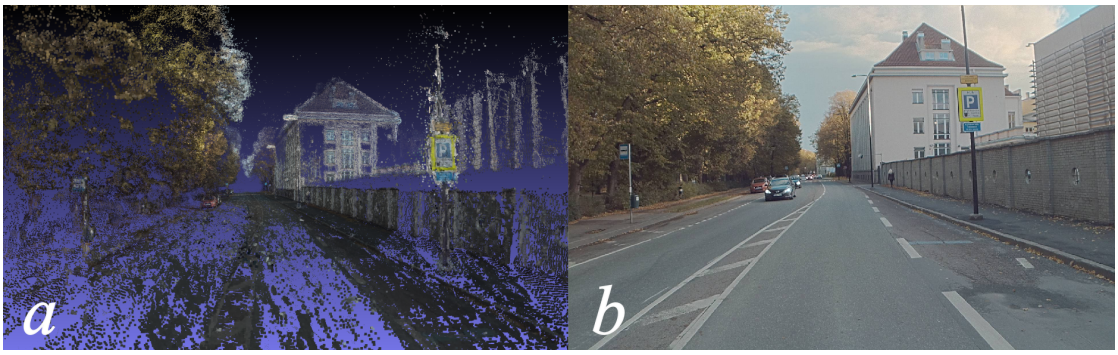


Figure 13. Dense point cloud created by OpenSfM (a) with a camera frame from the same position for reference (b).

It is certainly possible to create an additional component to perform real-time localization in the SfM-generated point cloud, but due to scope and time constraints, we decided against that and deemed SfM as a not viable option.

ORB-SLAM 3 In contrast to OpenSfM and SfM in general, ORB-SLAM 3 is a SLAM system, meaning it can be used for both mapping and localization in real-time. Additionally, it is open-source and regarded as state-of-the-art among SLAM systems in terms of accuracy and robustness [19, 27]. The ORB-SLAM family has numerous forks on GitHub, as well as an active following in the form of blog articles and published works based on it [14, 28].

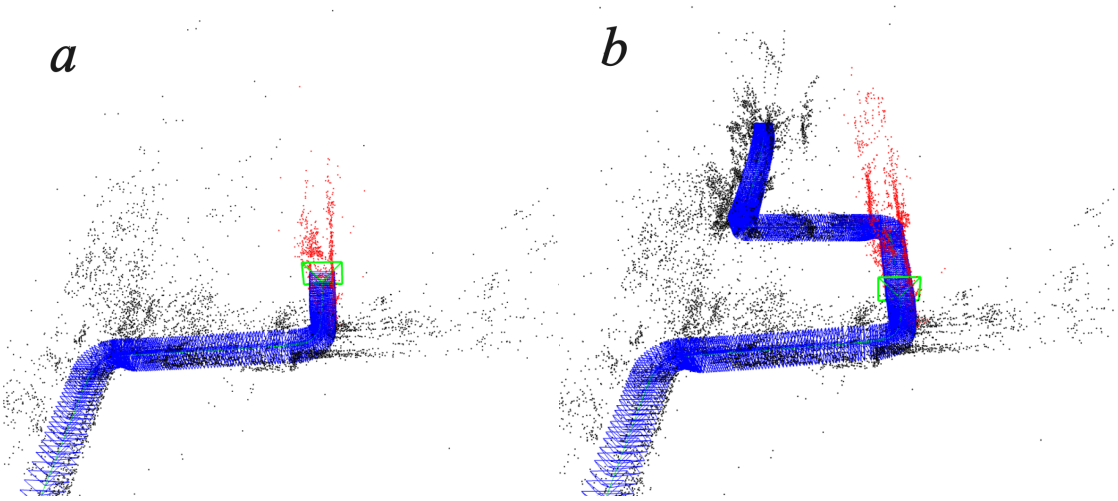


Figure 14. ORB-SLAM 3 in action: SLAM mode (a) and localization-only mode (b) using a previously created map.

ORB-SLAM 3 also supports various IMU and camera configurations, including monocular, monocular-inertial, stereo, stereo-inertial, and RGB-D. Our primary interest lies in the monocular and monocular-inertial configurations.

Additionally, ORB-SLAM 3 utilizes Atlas [5] for managing map data, which supports the storage of multiple disjoint maps, enables map merging in cases of overlap, and accommodates multiple sessions. This allows users to save an Atlas instance to a file, load it later, and continue updating it as needed. To our knowledge, no other available open-source SLAM systems offer such an advanced abstraction for map data management. While there are ORB-SLAM 2 implementations [16] that separate map storage by using external databases like PostgreSQL and PostGIS, ORB-SLAM 3, being based on ORB-SLAM 2, could potentially benefit from similar techniques. We believe that many of the insights gained from these implementations could be transferable, should we decide to adopt a similar approach.

Considering all these factors, we believe it better matches the requirements we set for the system and so we decided to proceed with ORB-SLAM 3 as the foundation for our system.

4.3 ORB-SLAM 3 Modifications

We implemented minor adjustments to ORB-SLAM 3 in order to incorporate GNSS data into its map data structures and provide additional output data.

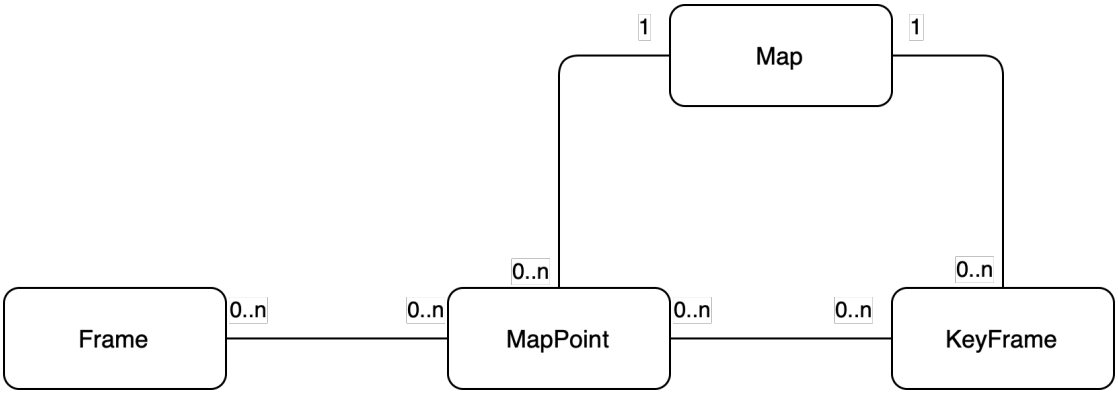


Figure 15. Key map data structures in ORB-SLAM 3 and their relationships.

Figure 15 shows the 4 key data structures that are used to represent map data in ORB-SLAM 3:

- *MapPoint* represents a 3D point in the SLAM coordinate system. It can be associated with multiple *KeyFrames* where it is visible.
- *Frame* is a single image captured by the camera together with associated data such as the camera pose and detected ORB features.
- *KeyFrame* is a special type of *Frame* that is selected based on certain criteria (e.g., the number of tracked *MapPoints*) and is used during mapping in BA and the loop closure process.
- *Map* is a container for *KeyFrames* and *MapPoints*.

We incorporated GNSS data into Frame and KeyFrame structures and utilized this information when saving SLAM trajectory data at the end of a run, which also includes the associated ground truth data. All trajectory data is stored in the TUM[7] format.

We run ORB-SLAM 3 in two modes: SLAM and localization. In the SLAM mode we generate the following files:

- *atlas.osa*: Binary serialization of Atlas, this includes all maps, key frames and map points. This file is loaded when ORB-SLAM is used in localization mode.
- *m_*_trajectory.txt*: Trajectory during mapping in SLAM coordinates, generated from the *KeyFrames* in the map and used for training the coordinate transformation model.

[7] `https://cvg.cit.tum.de/data/datasets/rgbd-dataset/file_formats`

- *m_*_trajectory_gt_wgs.txt*: Ground truth in WGS84 coordinates, used for visualization and for training the coordinate transformation model.

- *m_*_map_points.txt*: x, y, z coordinates of map points, used for visualization.

In the above file names, '*' represents the map index in Atlas. If ORB-SLAM 3 loses tracking and cannot relocalize, it will create a new map and build on that. If it cannot merge with a previous map, it will remain a disjoint map in Atlas. For training, we pick the longest map (most keyframes) and discard the rest.

In the localization mode localizing the following files are generated:

- *l_*_trajectory.txt*: Trajectory during localization in SLAM coordinates, generated from all tracked *Frames* and used as input for the coordinate transformation model.

- *l_*_trajectory_gt_wgs.txt*: Ground truth in WGS84 coordinates, used for visualization.

In this case, '*' indicates the sequence number of localization, and this number increments with each new localization run. It is worth noting that localization files with index 0 (*l_0_trajectory.txt* and *l_0_trajectory_gt_wgs.txt*) have a special meaning in our system. They are created by running ORB-SLAM 3 in localization mode on the same input data that was used for mapping. For brevity we will refer to this as "self -localization".

4.4 Training Data

We train and evaluate our model using three sequences that cover a range of challenging scenarios for a monocular visual SLAM system. These include sharp turns, loop detection, and varying viewpoints. In this work, we do not consider seasonal changes or extreme illumination differences, such as night/day transitions, since all sequences were recorded during daylight hours in the first half of October. Some example frames from input videos can be seen in Figure 20. Figure 16 presents the sequences used in this work.

The sequences were recorded using our test vehicle and saved in the ROS bag format. This format allows us to store additional data, such as GNSS and IMU information, in a time-stamped manner, ensuring that each data point is associated with a specific time instance. Furthermore, it simplifies the process of feeding the video stream into ORB-SLAM 3, simulating a real-time usage scenario of the system.

For video input we use the topic */interfacea/link3/image/compressed*, with the video recorded at 1920x1080 resolution and 30 frames per second. For GNSS data we use the topic */novatel/oem7/inspva*, this is recorded at 50 Hz.

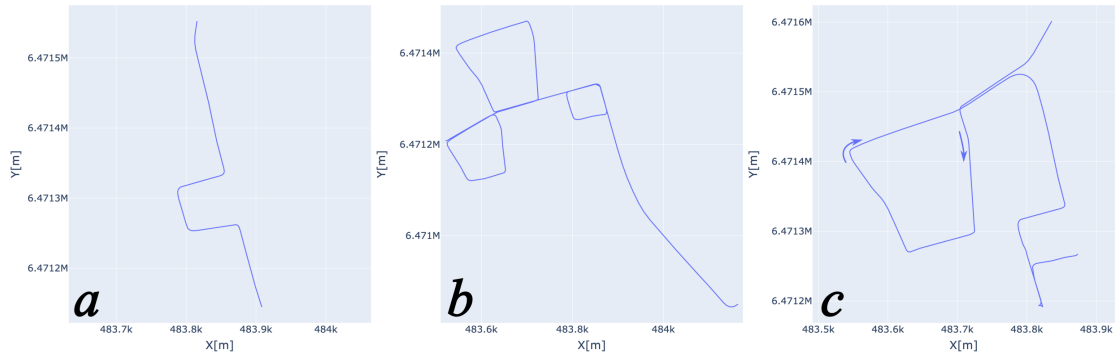


Figure 16. Ground truths of test sequences used in this work in UTM coordinates. From left to right Sequence 01 (a), Sequence 02 (b), and Sequence 03 (c).

We will briefly describe main challenges of each sequence:

- Sequence 01 has sharp turns and no loops, making it extremely difficult for monocular visual SLAM to avoid scale drift [22].
- Sequence 02 has multiple loops, the main challenge here for the SLAM system is to correctly perform loop closure.
- Sequence 03 presents a challenge in that it also has a loop but the loop is closed from a different direction than it is started (note the arrows in the rightmost panel in Figure 16).

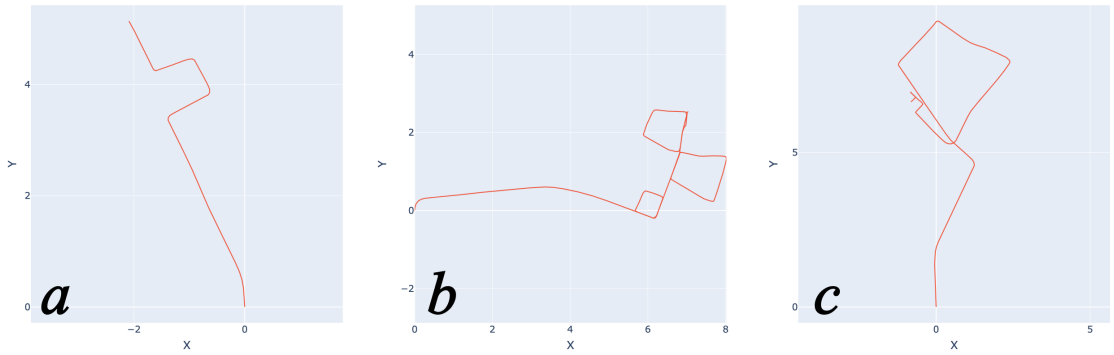


Figure 17. SLAM mapping trajectories for sequences 01 (a), 02 (b) and 03 (c) in SLAM coordinates, illustrating various challenges of monocular visual SLAM. Note the significant difference between the mapped trajectory in panel (c) and the ground truth in Figure 16 (c).

Figure 17 highlights the challenges of monocular visual SLAM on our test sequences. Observe the differences between the trajectories in Figure 17 and the ground truths in Figure 16, particularly in terms of orientation, scale, and overall shape. These discrepancies are most noticeable in cases where there is no loop closure or when loop detection fails. In Figure 17 (c) Sequence 03 serves as a prime example, where the loop is not accurately detected due to different viewpoints, leading to accumulated scale drift.

It is important to note that the scale and orientation of the SLAM trajectories are arbitrary, determined by the ORB-SLAM 3 tracking initialization process and unrelated to the actual real-world values. Consequently, the distortions in relation to the ground truth exhibit non-linear characteristics in the case of visual SLAM. Therefore, a linear transformation like Umeyama [29] is insufficient to map SLAM coordinates to real-world coordinates. To address this issue, we decided to employ a neural network model as we believe it would provide better results and generalize more effectively to unseen data compared to machine learning-based methods.

Test Data Generation As mentioned earlier, ORB-SLAM 3 can be run in either SLAM mode or localization-only mode. To achieve a high-quality map, we extract 2000 ORB features per frame and downscale the input video to 640x400 resolution. This improves system performance and ensures continuity during tracking.

To generate all the data required to train a model for a particular sequence and evaluate it, we follow these steps:

1. Run ORB-SLAM 3 in SLAM mode on the sequence video to create an Atlas map for the sequence and generate the trajectory files.

2. Perform self-localization by running ORB-SLAM 3 in localization mode on the original video and generating localization trajectory files.

3. Run ORB-SLAM 3 in localization mode on any overlapping sections from other sequences using the Atlas map created in step 1 and generate additional localization trajectories for evaluation.

4.5 Georeferencing using coordinate transformation

In our solution, georeferencing is done using coordinate transformation and is implemented as a separate post-processing step outside of ORB-SLAM 3. Our solution is built using the Keras deep learning framework [3] and we use KerasTuner [23] to find the best model topology and hyperparameters. We train a FCNN model to predict real-world coordinates from localized SLAM coordinates. As shown in Equation 1, the coordinate transformation model takes a point in SLAM coordinate space and outputs a point in UTM coordinate space.

$$T : (x, y, z) \mapsto (x', y', z') \tag{1}$$

4.5.1 Training

A separate model is trained for each mapped trajectory, which means that a model is specific to a certain area. Due to limited data, we train the model by fitting the SLAM trajectory obtained during mapping to the ground truth and use self-localization data for validation, which helps improve the model's generalization capability. The difference between the mapping trajectory and self-localization can be seen in Figure 18. Although they are generated from exactly the same input video and share exactly the same ground truth, there are slight differences in how ORB-SLAM 3 localizes the camera during mapping compared to localizing. This could be attributed to ORB-SLAM 3 picking slightly different features for triangulation when in localization mode due to an initialization difference.

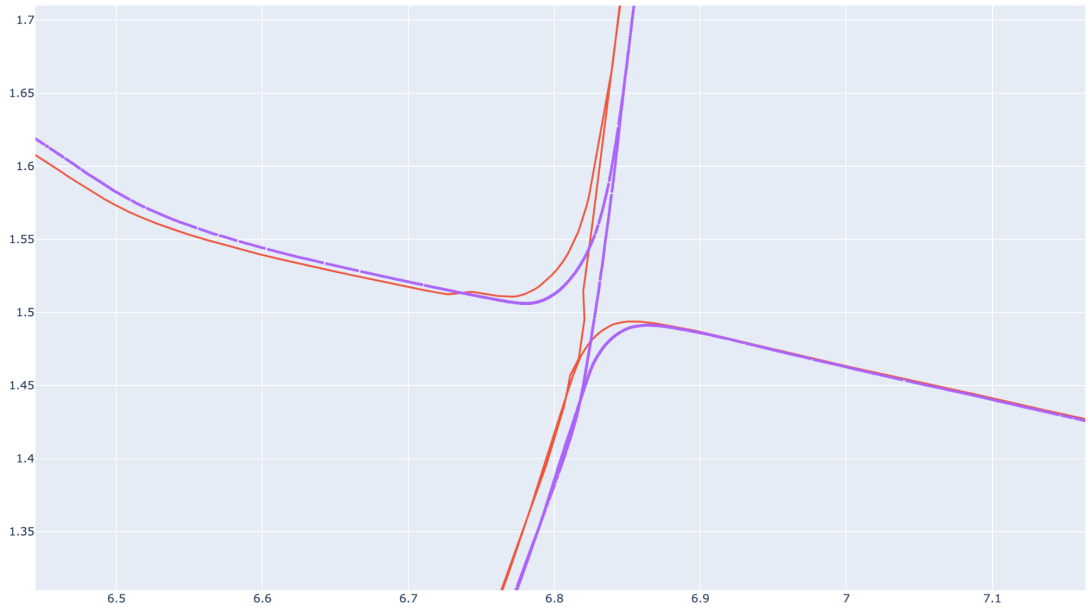


Figure 18. Mapping trajectory (red) compared to self localization trajectory (violet) in SLAM coordinates.

Training data is normalized using Keras' Normalization layer, which shifts and scales inputs into a distribution centered around 0 with standard deviation of 1. Since source (SLAM coordinates) and target data (UTM coordinates) are in different scales, we have two normalizer instances - one for source data and the other for target data. These normalizers are stored and reused later when doing predictions based on the model.

In our model, we use the Adam optimizer and the Huber loss function, which is less sensitive to outliers than mean squared error (MSE) and has demonstrated good results in our experiments. We set the maximum number of epochs for training to 200 and use early stopping together with adaptive learning rate that is reduced when the training plateaus. Table 1 shows a summarized view of these parameters, and Figure 19 shows loss graphs.

Table 1. Summarized view of model training parameters.

Parameter	**Comment**
Max epochs	200
Optimizer	Adam
Loss function	Huber
EarlyStopping	*patience=15*
ReduceLROnPlateau	*factor=0.2, patience=5, min_lr=1e-7*

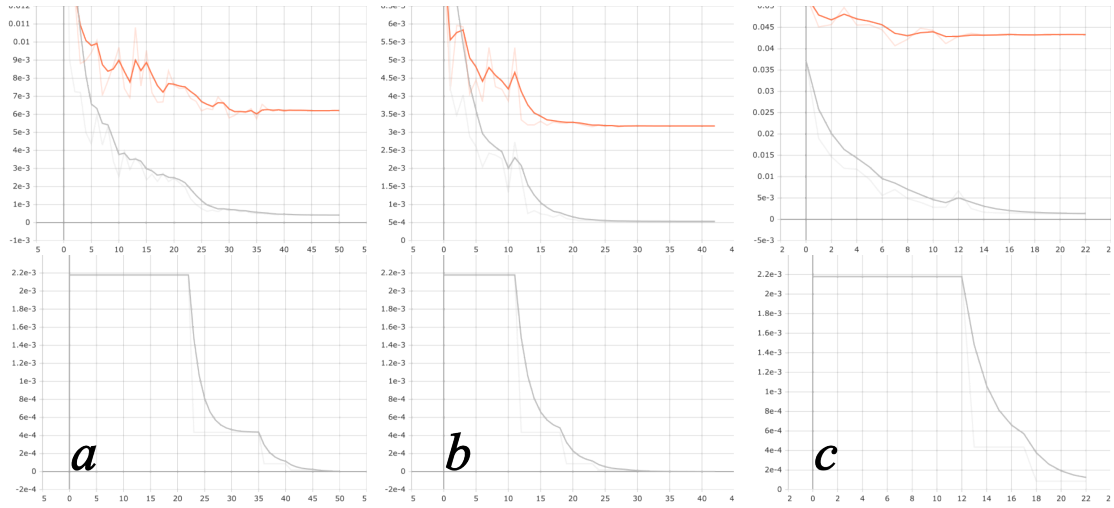


Figure 19. Training (gray) and validation (orange) losses (top row), along with learning rate (bottom row) for models corresponding to Sequence 01 (a), Sequence 02 (b), and Sequence 03 (c). The horizontal axis represents the epoch number.

4.5.2 Hyperparameter Search

For hyperparameter tuning, we utilize the Hyperband tuner from Keras with the primary objective of minimizing validation loss. Although we experimented with minimizing the Euclidean distance between true and predicted results, optimizing for validation loss has shown to yield better results. To speed up the search process, we uniformly downsample the training data by 50%. This approach preserves the core characteristics of a trajectory relevant to evaluating model performance while significantly reducing computation time compared to using the full dataset. Summary of the hyperparameter search space is shown in Table 2.

All models and their hyperparameters are inlcuded in `https://github.com/binaryunary/gorbslam`.

Table 2. Hyperparameter search space overview.

Hyperparameter name	**Value range**	**Comment**
il_units	[16..1024]	Number of nodes in input layer (first hidden layer)
il_activation	[relu, tanh]	Input layer activation function
hl_num	[0..5]	Number of hidden layers
hl_{i}_units	[16..1024]	Number of nodes in ith hidden layer
hl_{i}_activation	[relu, tanh]	Activation function of ith hidden layer
huber_delta	min=0.1, max=1.0, step=0.1	Delta parameter for the Huber loss function
learning_rate	min=1e-5, max=1e-2	Argument for the optimizer

4.6 Challenges

In this section, we will discuss the significant challenges encountered during the development and implementation of our proposed system. We will describe the strategies employed to address these challenges and analyze the implications of any insurmountable obstacles on our work. The primary challenges are related to ORB-SLAM 3 and include: IMU integration, initialization when localizing and real-time performance in general.

4.6.1 IMU Integration

Initially, our goal was to develop a mono-inertial system to achieve optimal SLAM performance within the scope of our requirements. The authors of ORB-SLAM 3 observed that mono-inertial systems offer only slightly lower accuracy compared to stereo-inertial configurations, which demonstrated the best performance in their study. However, despite multiple attempts, we were unable to integrate our IMU into ORB-SLAM 3. The authors also noted that initializing IMU sensors can be challenging in scenarios with slow motions or without roll and pitch rotations, such as a car driving in a flat area. Nevertheless, we believe this issue is not overly significant, as we will use ground truth data for training. Any scale drift or other mapping errors should be corrected through this process. As previously mentioned, the proposed system will train a model specific to a particular trajectory, meaning that these corrections are not generalizable to any arbitrary location but are tailored to a specific map and model.

4.6.2 Unreliable Initialization in Localization Mode

While ORB-SLAM 3's initialization is fast and robust in SLAM mode, its performance in localization mode is not as reliable, occasionally failing under conditions that appear similar to those encountered during mapping. Often, we have to replay a video multiple times before the system can successfully initialize and begin tracking existing features.

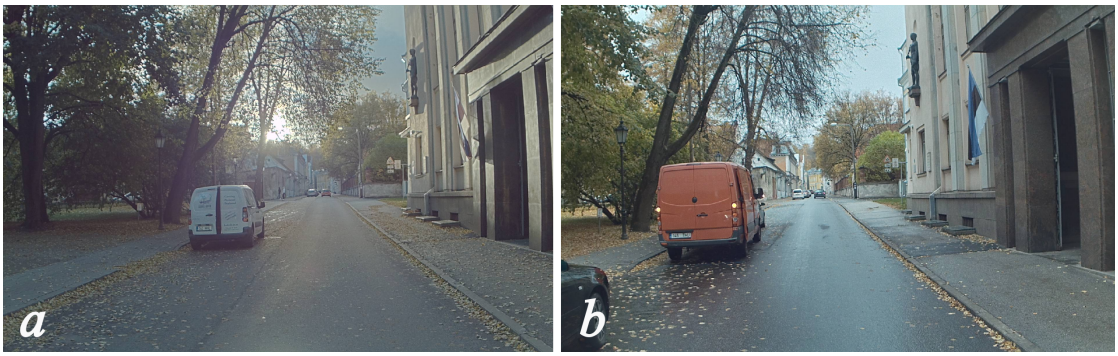


Figure 20. Examples of similar frames from overlapping sections in sequences 01 (a) and 03 (b). The SLAM map was created using sequence 01, localization was attempted on the overlapping section of sequence 03. Although the conditions appear similar, ORB-SLAM 3 failed to initialize and localize on sequence 03.

The initialization failure depicted in Figure 20 could potentially be attributed to illumination differences being significant enough between the two videos that not enough ORB feature matches could be found during localization. This is further exacerbated by the non-determinism of ORB-SLAM 3, possibly due to multithreading and the use of the RANSAC algorithm. This could also explain why replaying a video multiple times can yield different results.

4.6.3 Real-Time Performance

We use a custom ROS node to interface with ORB-SLAM 3. To simulate a real-world usage scenario where our system must work in real time, we set up subscriptions to relevant data topics with a queue size of one. This ensures that any skipped frames are not buffered and are instead dropped, similar to how the system would operate in a real-time.

The input video is captured at a 1920x1080 resolution, which needs to be downscaled for ORB-SLAM 3 to process it in real time. We were unable to achieve satisfactory performance on our development hardware, an Apple MacBook Pro with the M1 Pro CPU. We experimented with resolutions of 960x604, 640x400, and 480x300, and determined that 640x400 delivers the best and most consistent results. This resolution maintains

a balance between real-time operation and retaining enough information for feature extraction. With other resolutions, the system often lost track and was unable to relocalize. For instance, with the resolution of 960x604, the system was able to fully map Sequence 03 once out of 5 consequtive runs. With the resolution of 640x400, the system was able to do it every time.

Our experience trying to achieve real-time performance led us to consider that known benchmark datasets such as KITTI [8] are distributed as files and are consumed by the benchmarked systems by simply iterating over the files. This approach removes any real-time constraints and ensures optimal continuity. We believe the results of such benchmarks could be communicated more clearly by indicating that these results are not obtained under real-time conditions.

5 Evaluation and Results

This section is dedicated to evaluating the proposed method and analysis of the results obtained. In addition, we will compare the performance of the the proposed method against other machine learning based methods, such as Gradient Boosting Regression (GBR), Random Forest Regression (RFR), Support Vector Regression (SVR), and a naive linear transformation such as Umeyama.

We will first outline the evaluation process, then present the results of our experiments, and finally, we will discuss the implications of these findings and identify potential areas for future work and improvements.

5.1 Evaluation

As mentioned in Section 4.4 (Training Data), we use three sequences (Figure 24) to train individual Fully Connected Neural Network (FCNN) models for each sequence. During the evaluation, we perform SLAM localization using the Atlas map specific to that sequence. Overlapping sections (Figure 21) from other sequences are also localized against the same Atlas map using SLAM features. We then transform the SLAM coordinates to real-world coordinates using the corresponding FCNN model. The performance of each model is evaluated by comparing the FCNN predictions to the ground truth using the Absolute Position Error (APE) metric, which focuses on the position component while discarding rotation. We use the evo tool [10] to calculate APE. For more details about mapping and data preparation, please refer to Section 4.4.

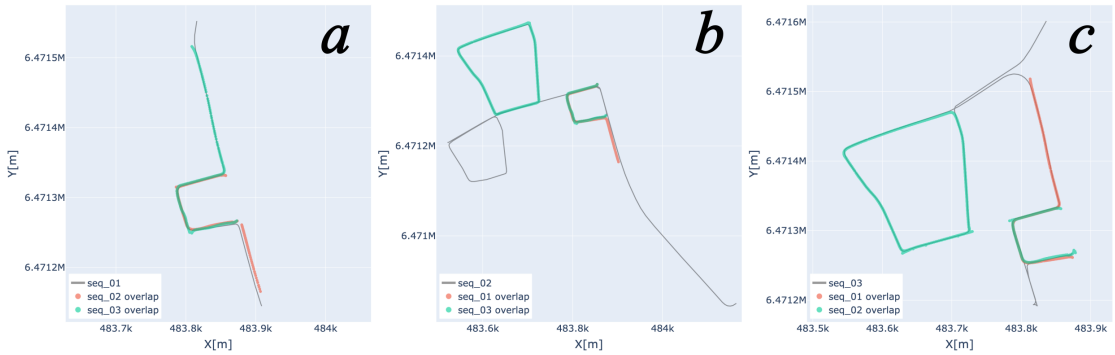


Figure 21. Ground truth trajectories in UTM coordinates with overlapping sections from other sequences highlighted for enhanced clarity. Panel (a): Sequence 1, including overlapping sections with Sequences 02 and 03. Panel (b): Sequence 2, including overlapping sections with Sequences 01 and 03. Panel (c): Sequence 3, including overlapping sections with Sequences 01 and 02.

For added clarity, we want to emphasize that the experiments are conducted using a ROS node to simulate the real-time behavior of the system, as it would be utilized in real-world scenarios. Running the system in real-time also highlights the localization initialization issue of ORB-SLAM 3, which was discussed in Section 4.6.2 (Unreliable Initialization in Localization Mode) and is illustrated in Figure 22. This is further highlighted by the fact that for Sequence 01 ORB-SLAM 3 was unable to find enough feature matches to initialize and localize using overlapping sections that can be seen in Figure 21.

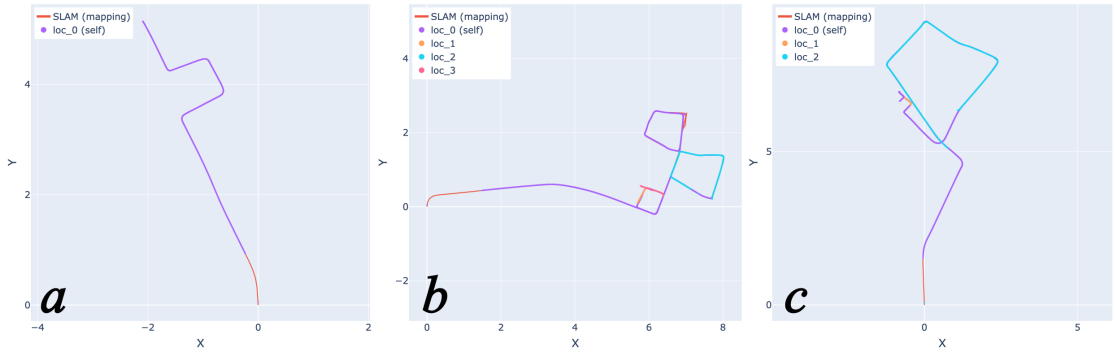


Figure 22. SLAM localization results per each test sequence. Note the ORB-SLAM 3 initialization delay that can be observed in self-localization; the purple track (self-localization) starts later than the red track (SLAM mapping trajectory).

5.2 Performance

To evaluate and validate the performance of our FCNN model, we compare it to several other Machine Learning (ML) methods used for regression problems: Gradient Boosting Regression (GBR), Random Forest Regression (RFR), and Support Vector Machine(SVM). We used the implementations of these algorithms provided by the Scikit-learn package [24]. For completeness, we also included a linear transformation algorithm, Umeyama, to demonstrate its limitations in addressing non-linear mapping errors such as scale drift. For the ML methods, we conducted a hyperparameter search to find the best parameters.

The results are summarized in Table 3, where we compare the mean APE statistics for each method across the sequences. It can be observed that the FCNN model achieves better localization accuracy (lower mean APE) for all sequences except Sequence 01. Sequences 02 and 03 have multiple localization trajectories and contain challenging features in the training data, such as loops and scale drift. The fact that FCNN managed to achieve the lowest APE suggests that the FCNN-based method is more generalizable.

In addition to a lower mean APE, the FCNN method has a lower mean APE standard deviation, indicating that this model produces more stable results.

Table 3. Summary of results for each method per sequence, ordered by the mean APE averaged across all localizations within that sequence.

Sequence	**Model**	**Mean of APE mean**	**Mean of APE median**	**Mean of APE RMSE**	**Mean of APE STD**	**no. of localizations**
seq_01	GBR	0.47	0.37	0.72	0.55	1
	FCNN	0.55	0.47	0.65	0.36	1
	RFR	0.60	0.45	0.80	0.54	1
	SVR	6.25	6.20	6.89	2.90	1
	UMEYAMA	15.40	13.66	19.21	11.49	1
seq_02	FCNN	2.05	1.74	2.37	1.17	4
	GBR	3.91	2.84	5.38	3.57	4
	RFR	4.05	2.30	9.17	7.91	4
	SVR	9.09	9.66	9.62	3.15	4
	UMEYAMA	23.24	22.05	24.51	6.43	4
seq_03	FCNN	1.95	1.76	5.53	5.06	3
	RFR	2.19	1.14	7.74	7.41	3
	GBR	12.58	10.89	16.71	10.68	3
	SVR	18.16	11.06	26.28	18.86	3
	UMEYAMA	58.68	55.01	61.83	18.67	3

In contrast, Sequence 01 has a relatively simple geometric shape and only one localization trajectory (self-localization). The top three models, GBR, FCNN, and RFR, exhibit similar results for this sequence. We attribute this similarity to the simple shape of Sequence 01 and the lack of localization trajectories from overlapping sections of other sequences, thus not fully testing the generalizability of the model.

If we consider APE across all sequences the FCNN model clearly presents better results as can be seen in Table 4. Not only does it produce more accurate results but the results are also more stable and the range of variation is considerably smaller than with other methods, which can be seen in Figure 23.

Table 4. APE (m) statistics by model across all sequences

Model	Mean of APE mean	Mean of APE median	Mean of APE RMSE	Mean of APE STD
FCNN	1.83	1.59	3.34	2.53
RFR	2.92	1.63	7.59	6.80
GBR	6.73	5.55	9.05	5.86
SVR	12.13	9.75	15.53	9.01
UMEYAMA	35.55	33.36	37.84	11.65

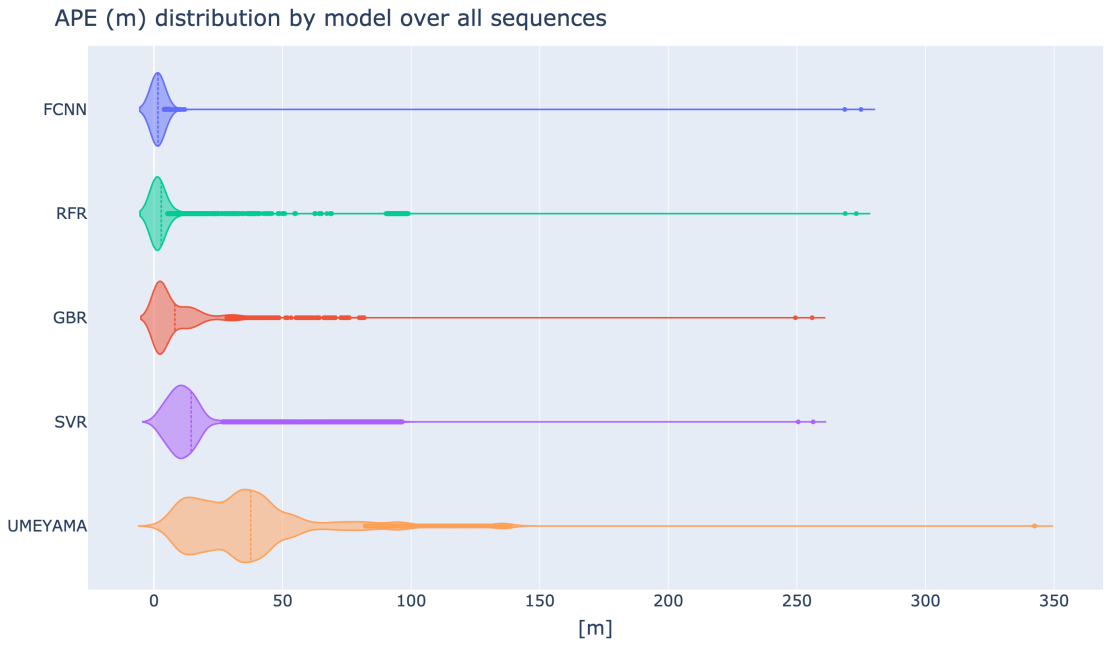


Figure 23. Violin plot of APE distributions for different models. The dashed line represents the mean value. Note the relatively small APE variability for the FCNN model compared to the other models.

To better visualize the performance differences between various methods, we present the results of all models on the longest sequence, Sequence 02, in the order of their performance: FCNN (Figure 25), GBR (Figure 27), RFR (Figure 26), SVR (Figure 28), and Umeyama (Figure 29).

In addition, to showcase how different models peform in case of severe scale drift in Sequence 03, we present the results of FCNN (best) and RFR (second best), and Umeyama (worst) in Figures 30, 31 and 32 respectively.

All of the figures referenced above can be found in Appendix II. Figures.

Color grading is used to highlight the APE values, with red representing the highest APE values and blue the lowest. The scale is the same for subfigures of the same sequence, but not across sequences. To avoid outliers skewing the color scale, we clip the APE values to the 99th percentile of the APE distribution for each sequence. The complete set of figures with all model and sequence combinations can be generated and explored interactively using the provided Jupyter notebooks in `https://github.com/binaryunary/gorbslam`.

6 Discussion

In this section, we discuss the findings from the evaluation and results presented in Section 5. We analyze the performance of our proposed method and its implications for real-world applications. This discussion aims to provide insights into the strengths and weaknesses of the proposed system and offer possible explanations for the observed results.

The initial results are promising. Despite having limited training and test data, our method achieves comparable localization accuracy to the best smartphone GPS[4], with an average APE RMSE of 3.34 m vs 3.28 m, and it surpasses the average APE RMSE across all tested phones of 9.59 m[4]. Given the wide range of smartphones used by ride-hailing drivers, our method has the potential to offer an improvement over the average smartphone GNSS receiver accuracy. We believe that our system could serve as a valuable addition to existing GNSS-based localization systems by compensating for some of their weaknesses.

Integrating both GNSS and our system could lead to a more robust and accurate localization solution. While GNSS systems may struggle in dense urban environments due to signal obstructions, our method can help overcome this limitation. On the other hand, our system relies on previously mapped areas, which makes it less suitable for unmapped regions or areas with challenging conditions for obtaining GNSS signals. By combining the strengths of both systems and addressing their respective weaknesses, an integrated solution could provide a significantly improved localization experience for users.

6.1 Limitations

The proposed system is designed for a fairly specific use case: on-road vehicles. Moreover, the constraints defined at the beginning of this study, particularly the requirement for the on-board component of the system to be entirely contained within the driver's device, impose certain limitations on the system's applicability. These constraints must be considered when planning future work and potential adoption.

The most significant limitation is that the proposed system can work effectively only if the area has been previously mapped and a coordinate transformation model trained. This makes it unsuitable for areas that have not been mapped or have challenging conditions for obtaining GNSS signals for mapping, such as tunnels and other underground structures. Furthermore, since the underlying tracking and mapping method is feature-based, feature detection is sensitive to viewpoint variation. For example, when traversing and mapping a trajectory in one direction, the obtained map and features may not be as useful when traversing the same trajectory in the opposite direction. Thus, it is necessary to map the area in both directions to be able to train a model that can handle bidirectional georeferencing effectively.

Additionally, monocular SLAM itself has several limitations that have to be taken into account. It is difficult for the system to regain tracking once it loses it due to either being unable to extract enough features from a frame because there is not enough texture, a problem inherent to feature-based odometries, or due to breaking continuity during pure rotations [22], e.g., steep (around 90 degrees or more) turning angles. This issue could potentially be remedied by post-processing the map data and stitching together the maps of different segments using ground truth data. Another problem inherent to monocular SLAM is scale drift, which could be solved by integrating an additional sensor, e.g., IMU. The effect of scale drift in the context of our system is not exactly known, but we can assume that if both the SLAM trajectory obtained during initial mapping and ground truth share a similar topology, the learning process would be more efficient and thus produce a better model.

Since the system is intended for on-road use, it is not expected to perform well for pedestrian use. Mapping is performed using dash-mounted smartphone cameras, so the features in the SLAM map will be observed from a different viewpoint compared to the pedestrian viewpoint, which is typically from a sidewalk, close to buildings, etc. Additionally, many pedestrian areas are off-limits for road vehicles, so they will remain unmapped.

Lastly, another notable limitation is the need for specialized equipment to obtain high-quality ground truth data for training the coordinate transformation model. Although high-precision GNSS receivers are small devices and their installation is not particularly challenging, this still means that not all vehicles in the fleet can be seamlessly incorporated into both the mapping and localization processes when the system is deployed to driver devices. We suggest potential adopters of this system install high-precision GNSS receivers into a small set of vehicles in their fleet. As the hardware is small and unobtrusive, these vehicles can still be used for regular passenger rides without issues.

6.2 Future Work

Considering that our current implementation serves as a proof of concept, we can categorize the future work into two main areas: advancing the research on data refinement and core concepts, and enhancing the implementation itself.

6.2.1 Core Concepts

More data is needed to better understand the performance characteristics of the proposed system. Figure 19 indicates a plateau in training and validation losses, which may suggest that the models have reached their learning capacity based on the training data currently available. This is somewhat expected, as we have only one mapping session per sequence (SLAM map and ground truth) to train a model. We expect the model to improve and

generalize better with more training data, especially when obtained while traversing the same trajectory in both directions, as mentioned in Section 6.1.

During the course of this thesis, we were unable to successfully integrate the IMU on our test vehicle with ORB-SLAM 3. While it is certainly feasible, more time is needed to debug the specific reasons behind the integration issues with ORB-SLAM 3. Obtaining a higher-quality SLAM trajectory that more closely aligns with the ground truth has less distortions related to scale drift , would aid in training a more accurate model.

To further enhance the training data and SLAM localization robustness, incorporating semantic segmentation of video input could be considered. This approach would enable the removal of dynamic noise, such as cars and pedestrians, from both the SLAM map and tracking processes. Tracking static features would increase the robustness and accuracy of the SLAM system.

As a side note, we observed during this thesis that all articles about SLAM systems primarily report the SLAM accuracy, while the subsequent localization on the created map seems to be less studied, making it difficult to compare localization performance specifically.

6.2.2 Implementation Related

This section focuses on improvements that would have to be made for our proposed system to be functional and deployable in a real-world scenario. We see two general high-level areas of improvement before our proposed solution could be considered for production development: central storage of SLAM maps and visual features to enable distributed access, and efficient management of SLAM maps together with FCNN models for coordinate transformation.

To construct a distributed system with shared SLAM maps, it would be necessary to decouple the Atlas mapping subsystem from ORB-SLAM 3 and integrate it into a separate component. This effort could potentially build upon the work accomplished in M2SLAM [16], where the authors move the internal tightly coupled map tracking component of an ORB-SLAM 2 system into a separate service that uses a Postgres database with PostGIS.

The current approach of having separate SLAM maps and models for each trajectory is not ideal, as it increases system complexity and storage requirements. A possible solution to mitigate this issue is to cluster individual maps based on geographic proximity, forming larger, more comprehensive maps. Models can then be retrained after the merging process to still provide correct coordinate transformations. An additional challenge is determining which model to use for coordinate transformation, whether this should be based on the driver device's GNSS position or visual features.

Lastly, as an alternative solution to the model management problem mentioned earlier, we would like to explore the possibility of integrating the GNSS data directly into the BA process of ORB-SLAM 3 during mapping and building the map in the UTM

coordinate system. This would eliminate the need for a separate training process and model altogether, as the SLAM map would already be in the correct coordinate system. This approach would also eliminate the need for a separate GNSS receiver for training, as the GNSS data would be obtained from the driver device. The main challenge with this approach is the integration of GNSS data into ORB-SLAM 3, which is not a trivial task.

7 Conclusion

In this work, we presented a proof of concept for a novel visual SLAM-based localization system that leverages a trained FCNN model to provide georeference for on-road vehicles. This system could be used in conjunction with GNSS-based systems for improved localization accuracy or as a fallback if GNSS signals are unreliable or unavailable. The study is primarily inspired by the poor localization accuracy of smartphone GNSS receivers, which ride-hailing operators predominantly use for localization and route planning.

Such inaccuracies lead to suboptimal route calculations for drivers and incorrect ETA for passengers, resulting in decreased customer satisfaction and potentially lost revenue. Our proposed system offers significantly higher localization accuracy than the average smartphone, with an APE RMSE of 3.34 m vs 9.59 m. We believe our system could be a valuable complement to existing GNSS-based localization systems. The two systems could be used in conjunction to provide more accurate localization results. In dense urban cores where GNSS signals are often obstructed by tall buildings and other structures, our system would be particularly effective. On the other hand, GNSS-based systems excel in open areas with clear line-of-sight to satellites and would be more reliable in scenarios where the vision-based SLAM system is affected by obstructed views or atmospheric conditions such as fog and heavy precipitation. By combining both systems, each system's strengths can compensate for the other's weaknesses, resulting in improved overall performance.

We believe these results are promising enough to justify continued research into the topics mentioned in the discussion. By refining and expanding upon this proof of concept, the system could eventually become a valuable addition to existing localization solutions for ride-hailing services and other on-road applications.

We have made the code related to this thesis publicly available to encourage further research and collaboration. You can find the relevant repositories at the following links:

- `https://github.com/binaryunary/ORB_SLAM3` - our ORB-SLAM 3 fork.
- `https://github.com/binaryunary/orbslam-dev` - Docker container for ORB-SLAM 3 development.
- `https://github.com/binaryunary/gorbslam` - coordinate transformation model together with test data used in this work and pretrained models.

References

[1] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José M M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate Open-Source library for visual, Visual–Inertial, and multimap SLAM. *IEEE Trans. Rob.*, 37(6):1874–1890, December 2021.

[2] Carlos Campos and Juan D Tardós. Scale-aware direct monocular odometry. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1360–1366. ieeexplore.ieee.org, October 2022.

[3] François Chollet and Others. Keras. `https://keras.io`, 2015.

[4] DXOMARK. GPS on smartphones: Testing the accuracy of location positioning. `https://www.dxomark.com/gps-on-your-smartphone-why-youre-not-always-there-when-it-says-youre-there/`. Accessed: 2023-4-16.

[5] Richard Elvira, Juan D Tardós, and J M M Montiel. ORBSLAM-Atlas: a robust and accurate multi-map system. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6253–6259. ieeexplore.ieee.org, November 2019.

[6] Florian Fervers, Sebastian Bullinger, Christoph Bodensteiner, Michael Arens, and Rainer Stiefelhagen. Uncertainty-aware vision-based metric cross-view geolocalization. November 2022.

[7] Yasutaka Furukawa, Carlos Hernández, et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

[8] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. *Int. J. Rob. Res.*, 32(11):1231–1237, September 2013.

[9] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[10] Michael Grupp. evo: Python package for the evaluation of odometry and SLAM.

[11] Brett Helling. How many uber drivers are there in 2023? `https://www.ridester.com/how-many-uber-drivers-are-there/`. Accessed: 2023-4-25.

[12] Ara Jafarzadeh, Manuel Lopez Antequera, Pau Gargallo, Yubin Kuang, and Torsten Sattler. CrowdDriven: A new challenging dataset for outdoor visual localization. September 2021.

[13] Doyeon Kim, Woonghyun Ga, Pyungwhan Ahn, Donggyu Joo, Sehwan Chun, and Junmo Kim. Global-local path networks for monocular depth estimation with vertical cutdepth. *arXiv preprint arXiv:2201.07436*, 2022.

[14] Dániel Kiss-Illés, Cristina Barrado, and Esther Salamí. GPS-SLAM: An augmentation of the ORB-SLAM algorithm. *Sensors*, 19(22), November 2019.

[15] Bryan Klingner, David Martin, and James Roseborough. Street view motion-from-structure-from-motion. In *2013 IEEE International Conference on Computer Vision*, pages 953–960. IEEE, December 2013.

[16] Fu Li, Shaowu Yang, Xiaodong Yi, and Xuejun Yang. Towards visual SLAM with memory management for Large-Scale environments. In *Advances in Multimedia Information Processing – PCM 2017*, pages 776–786. Springer International Publishing, 2018.

[17] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *Pattern Analysis and Machine Intelligence (PAMI)*, 2022.

[18] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. A comprehensive survey of visual SLAM algorithms. *Robotics*, 11(1):24, February 2022.

[19] Alexey Merzlyakov and Steve Macenski. A comparison of modern general-purpose visual slam approaches. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9190–9197. IEEE, 2021.

[20] Raúl Mur-Artal, J M M Montiel, and Juan D Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Rob.*, 31(5):1147–1163, October 2015.

[21] Raul Mur-Artal and Juan D Tardos. Visual-Inertial monocular SLAM with map reuse. October 2016.

[22] Raúl Mur-Artal and Juan D Tardós. ORB-SLAM2: An Open-Source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans. Rob.*, 33(5):1255–1262, October 2017.

[23] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, and Others. Keras tuner. *Retrieved May*, 21:2020, 2019.

[24] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau,

M Brucher, M Perrot, and E Duchesnay. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

[25] Lukas Platinsky, Michal Szabados, Filip Hlasek, Ross Hemsley, Luca Del Pero, Andrej Pancik, Bryan Baum, Hugo Grimmett, and Peter Ondruska. Collaborative augmented reality on smartphones via life-long city-scale maps. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 533–541. ieeexplore.ieee.org, November 2020.

[26] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[27] Dinar Sharafutdinov, Mark Griguletskii, Pavel Kopanev, Mikhail Kurenkov, Gonzalo Ferrer, Aleksey Burkov, Aleksei Gonnochenko, and Dzmitry Tsetserukou. Comparison of modern open-source visual SLAM approaches. August 2021.

[28] Jooeun Song and Joongjin Kook. Visual SLAM based spatial recognition and visualization method for mobile AR systems. *Applied System Innovation*, 5(1):11, January 2022.

[29] Umeyama. Least-Squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13:376–380, April 1991.

Appendix

I. Glossary

APE Absolute Pose Error.

BA Bundle Adjustment.

BEV bird's eye view.

CVGL cross-view geolocalization.

ETA Estimated Time of Arrival.

EXIF Exchangeable Image File Format.

FCNN Fully Connected Neural Network.

GBR Gradient Boosting Regression.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

IMU Inertial Measurement Unit.

ML Machine Learning.

ORB Oriented FAST and Rotated BRIEF.

RFR Random Forest Regression.

ROS Robot Operating System.

RTK Real Time Kinematic.

SfM Structure from Motion.

SLAM Simultaneous Localization and Mapping.

SVM Support Vector Machine.

UTM Universal Transverse Mercator.

WGS84 World Geodetic System 1984.

II. Figures

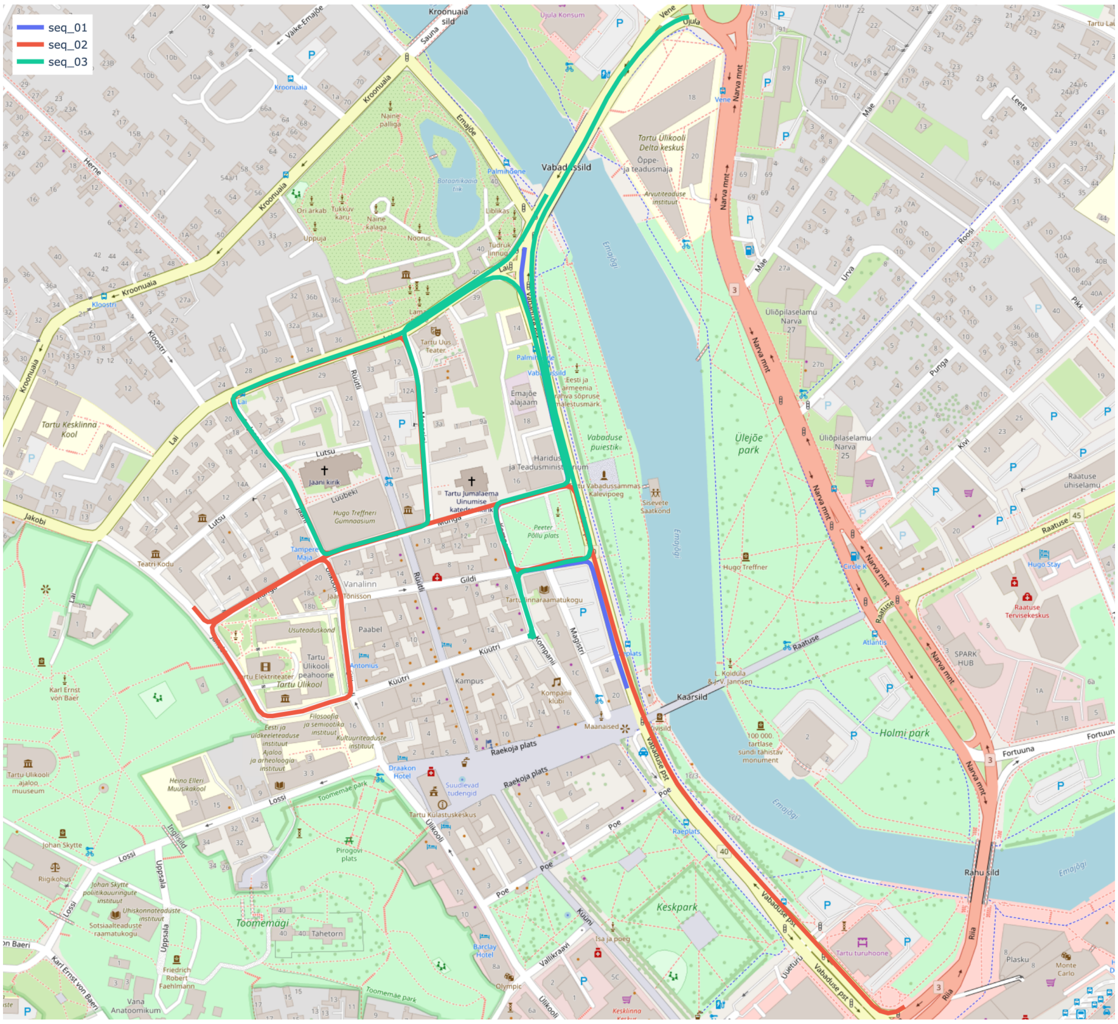


Figure 24. Sequences 01, 02, 03 plotted on the map of Tartu to provide additional spatial context and their relative positions to each other.

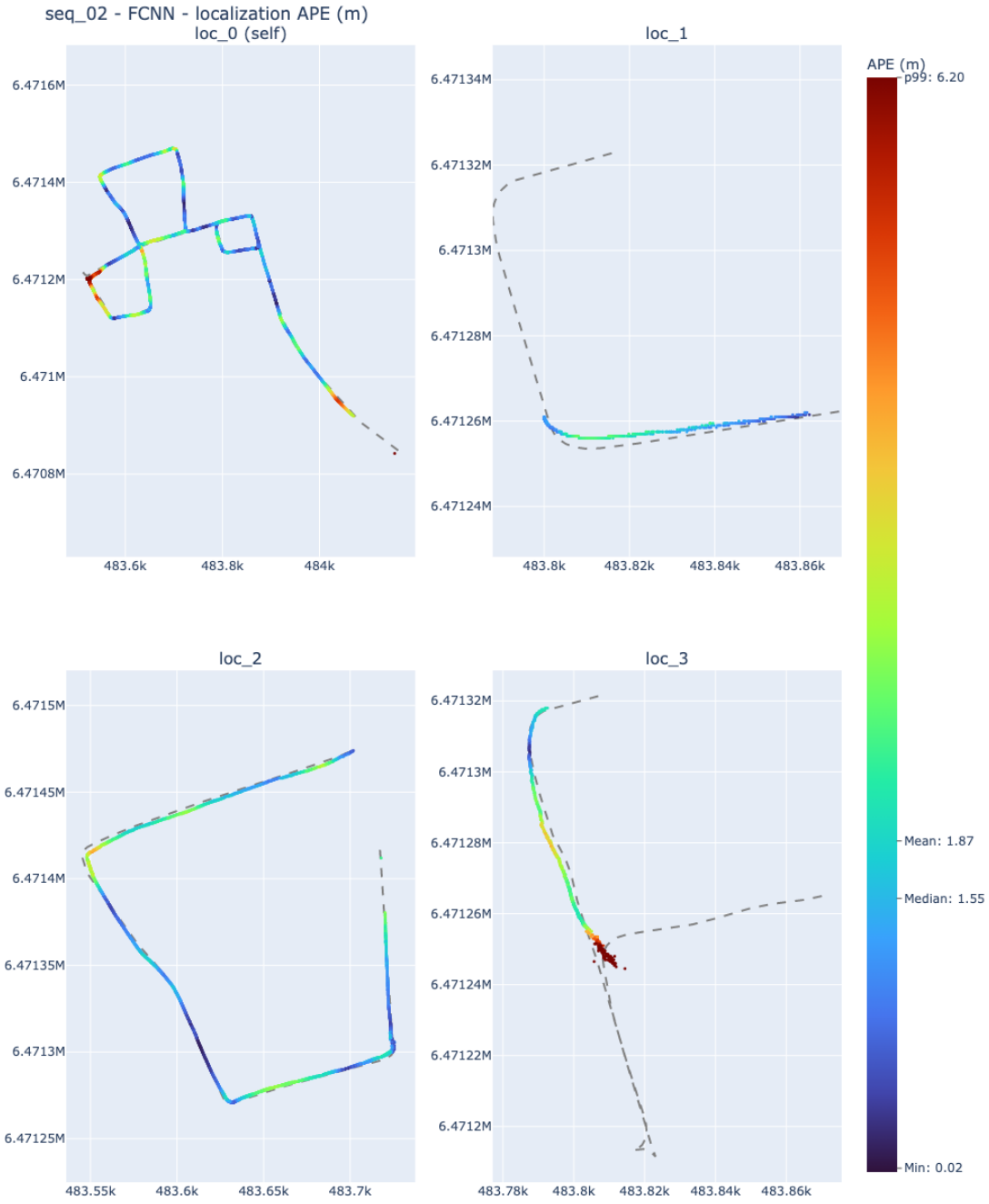


Figure 25. Colored APE plot of estimated trajectory vs. ground truth for FCNN on Sequence 02.

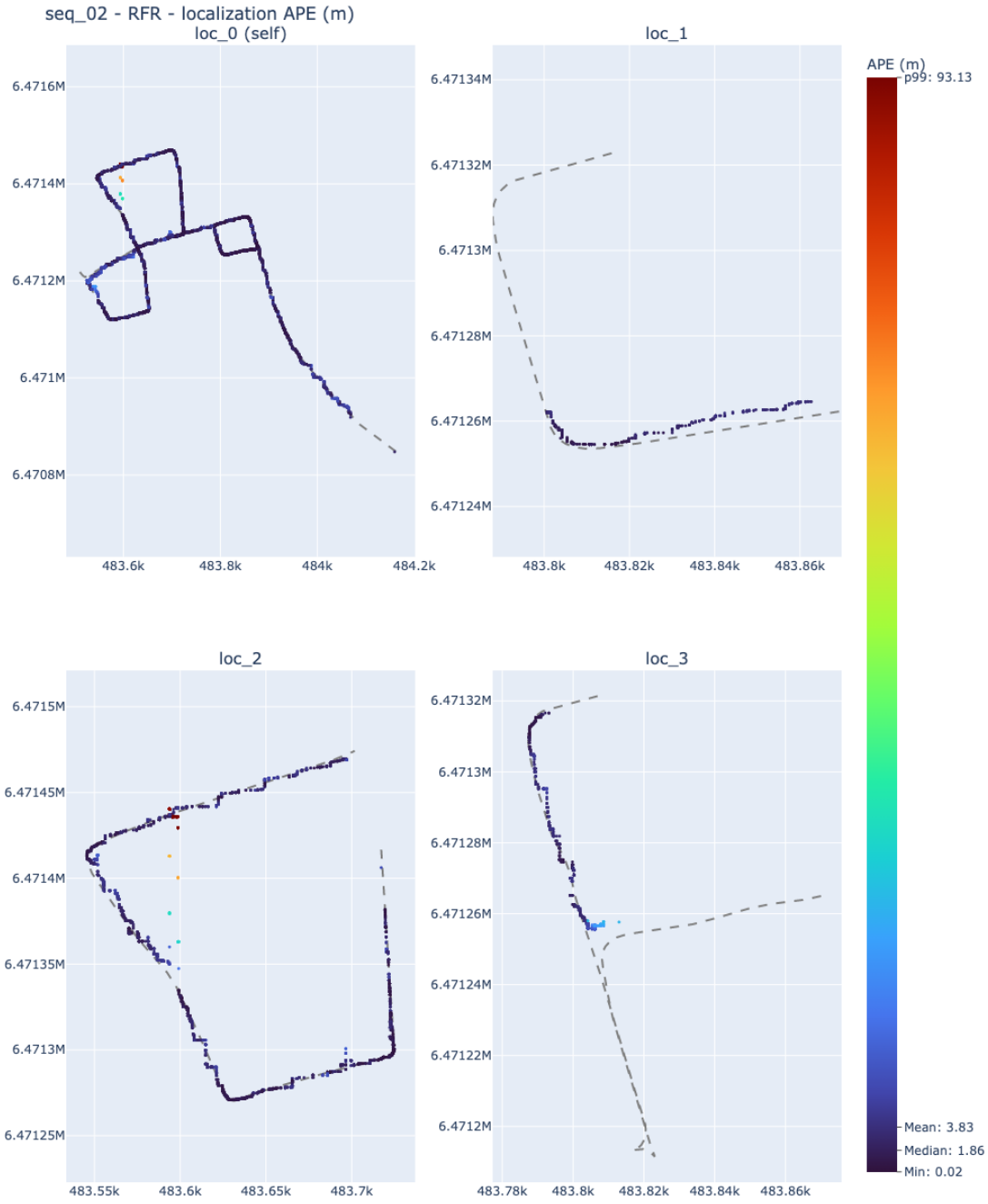


Figure 26. Colored APE plot of estimated trajectory vs. ground truth for RFR on Sequence 02.

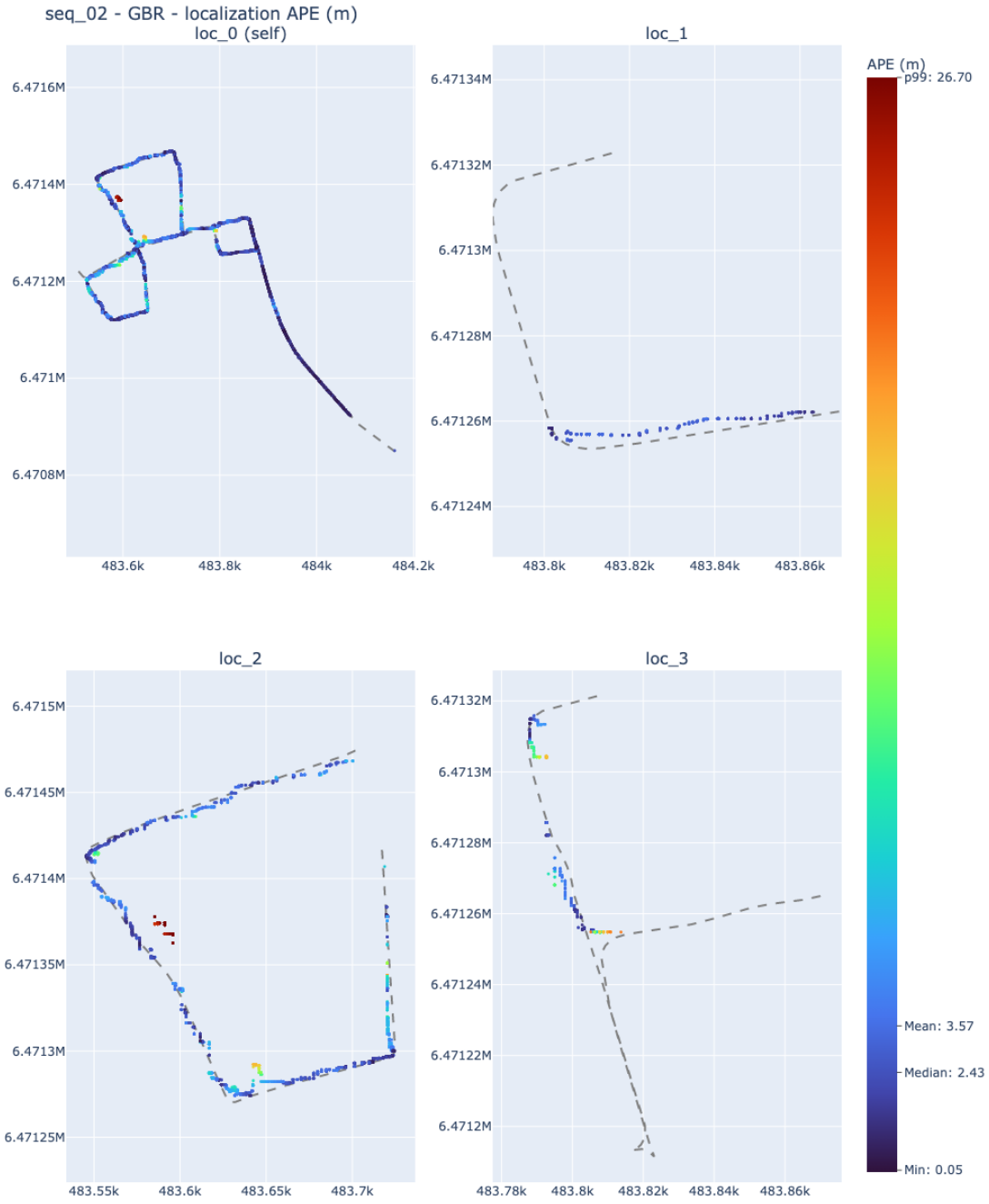


Figure 27. Colored APE plot of estimated trajectory vs. ground truth for GBR on Sequence 02.

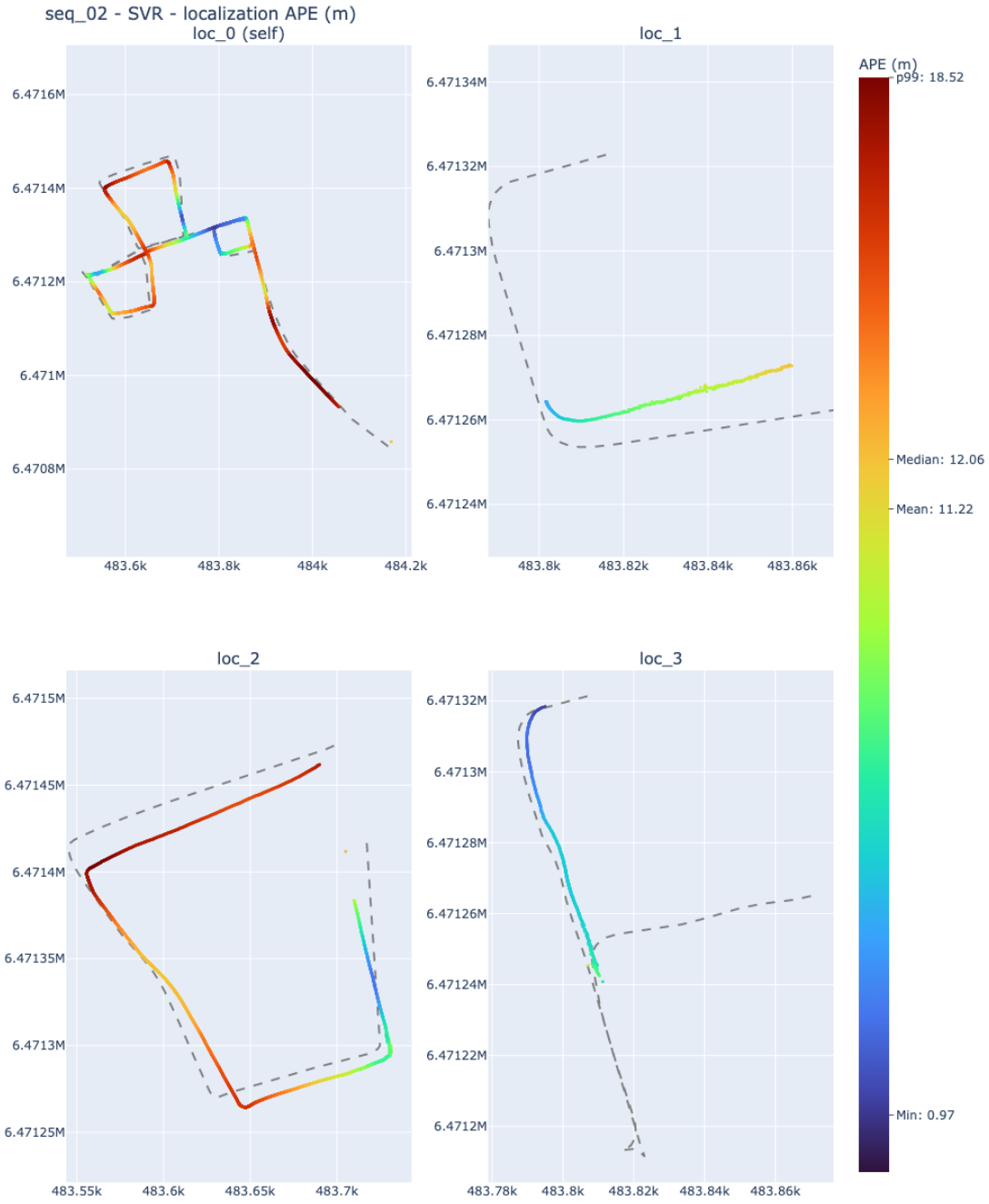


Figure 28. Colored APE plot of estimated trajectory vs. ground truth for SVR on Sequence 02.

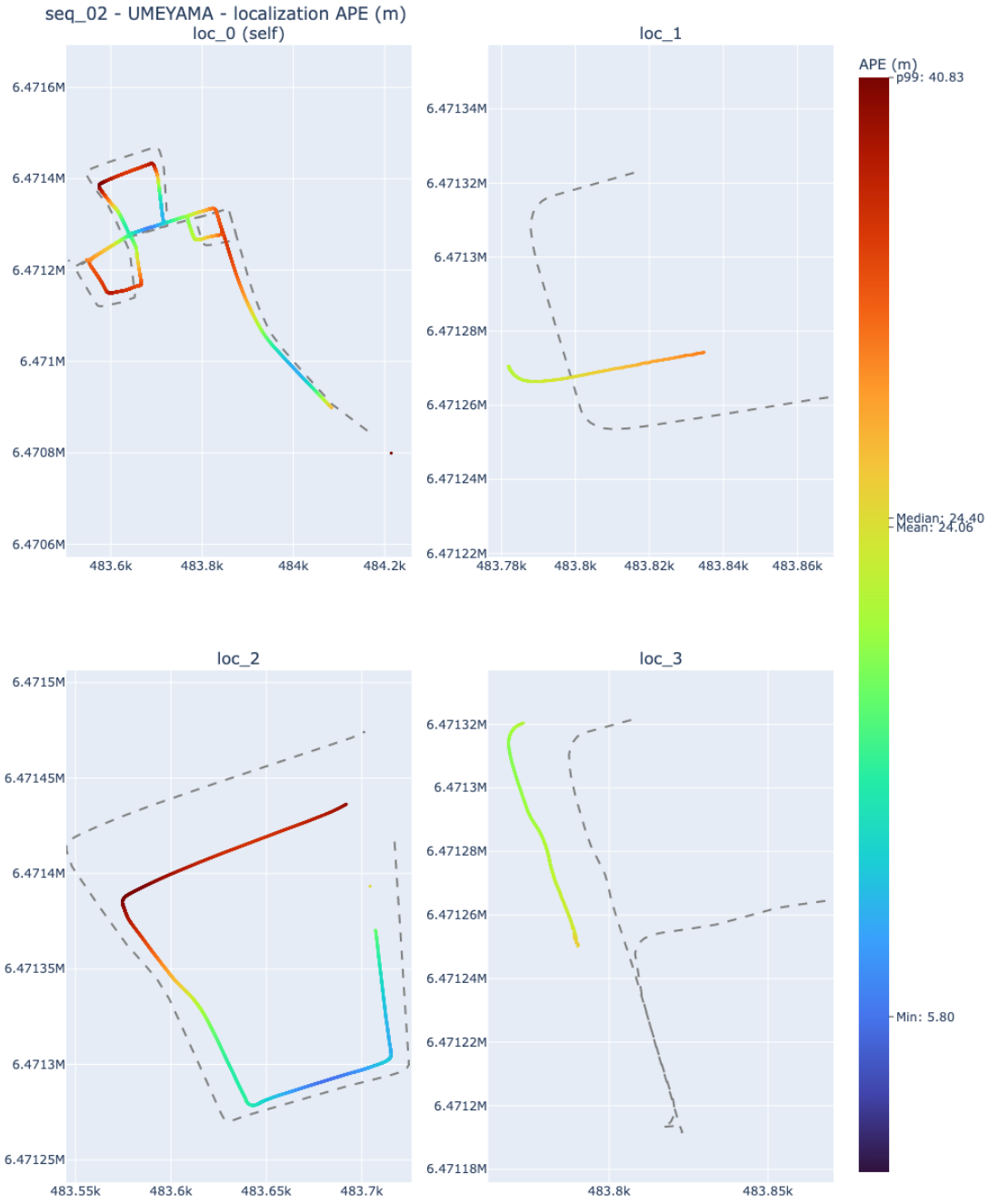


Figure 29. Colored APE plot of estimated trajectory vs. ground truth for Umeyama on Sequence 02.

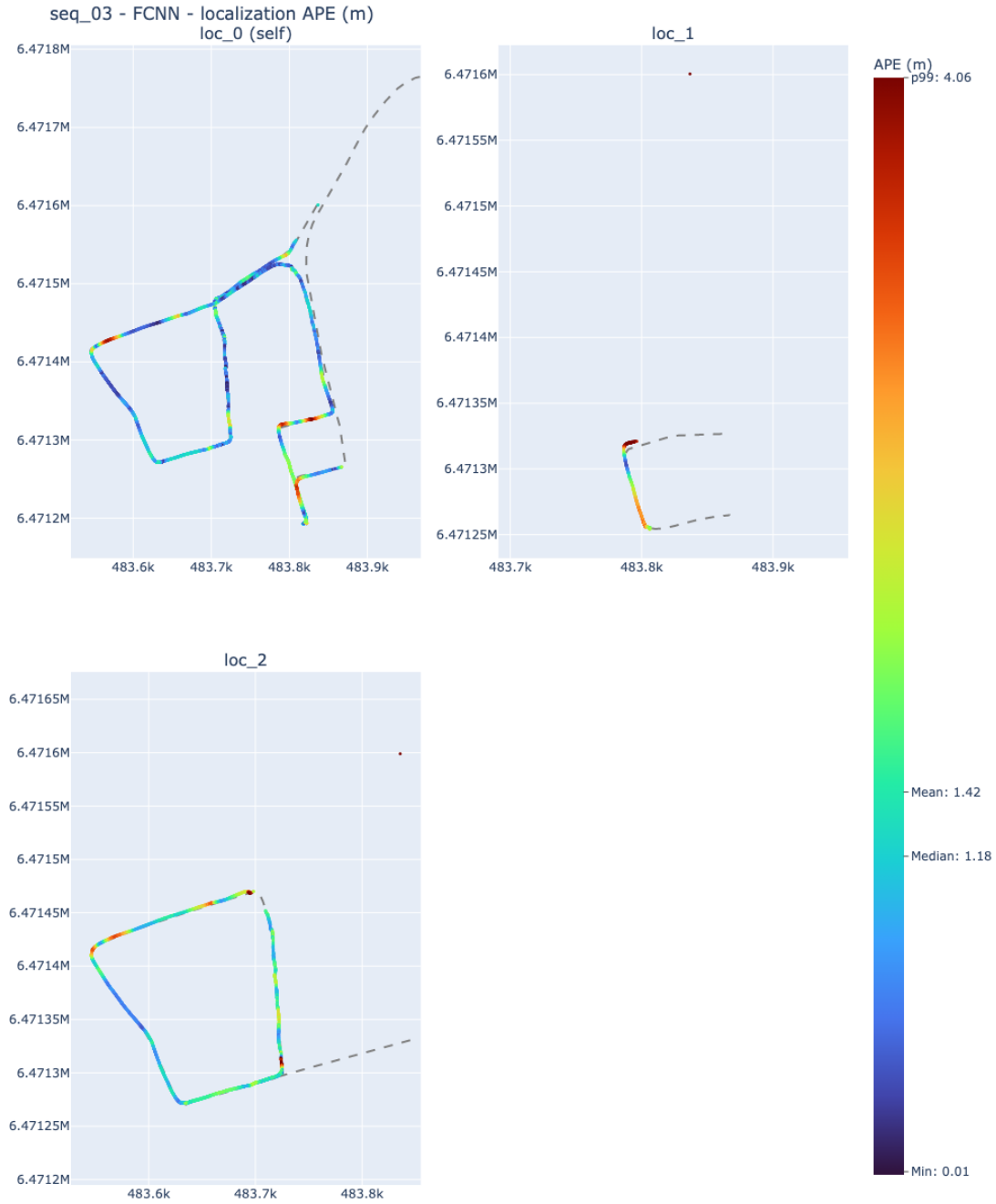


Figure 30. Colored APE plot of estimated trajectory vs. ground truth for FCNN on Sequence 03.

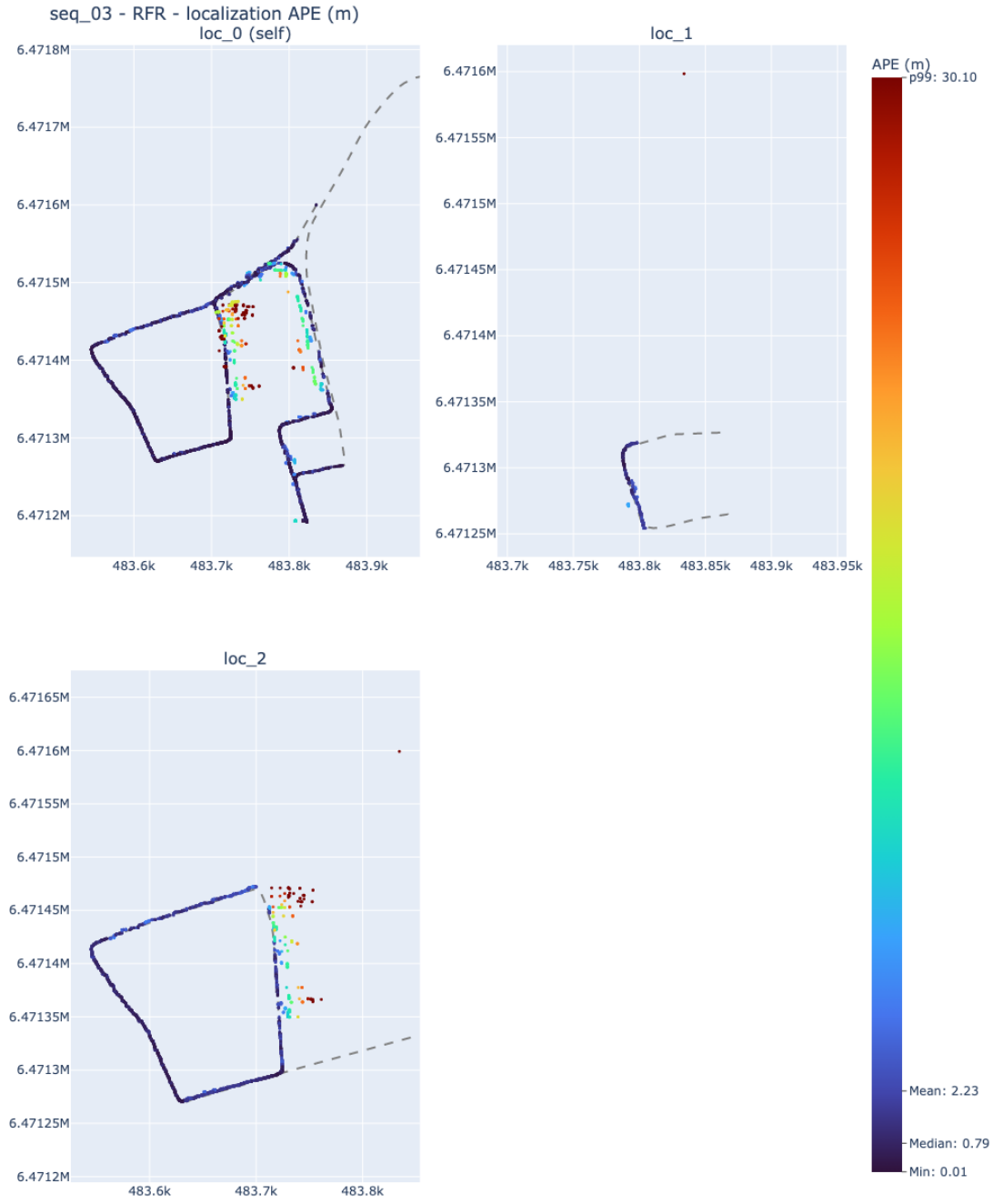


Figure 31. Colored APE plot of estimated trajectory vs. ground truth for RFR on Sequence 03.

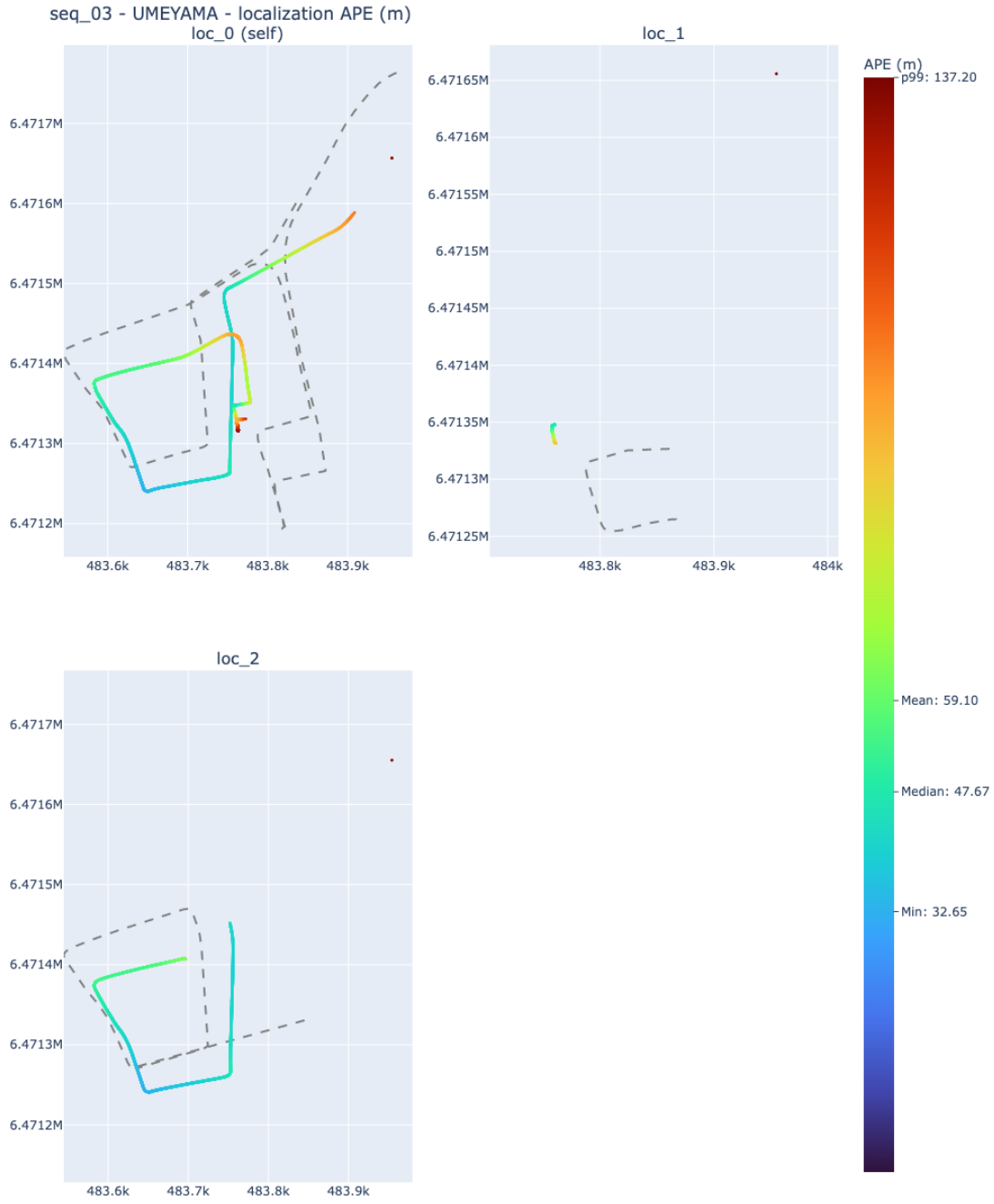


Figure 32. Colored APE plot of estimated trajectory vs. ground truth for Umeyama on Sequence 03.

III. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Erik Mägi**,
(author's name)

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,

   Georeferenced Visual SLAM,
   (title of thesis)

   supervised by Tambet Matiisen and Edgar Sepp.
   (supervisor's name)

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.

Erik Mägi
09/05/2023