PROPRIOCEPTIVE LOCALIZATION FOR ROBOTS

A Dissertation

by

HSIN-MIN CHENG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Dezhen Song
Committee Members,	Suman Chakravorty
	Dylan Shell
	Radu Stoleru
Head of Department,	Scott Schaefer

May 2021

Major Subject: Computer Science

ABSTRACT

Localization is a critical navigation function for mobile robots. Most localization methods employ a global position system (GPS), a lidar, and a camera which are exteroceptive sensors relying on the perception and recognition of landmarks in the environment. However, GPS signals may be unavailable because high-rise buildings may block GPS signals in urban areas. Poor weather and lighting conditions may challenge all exteroceptive sensors. In this study, we focus on proprioceptive localization (PL) methods which refer to a new class of robot egocentric localization methods that do not rely on the perception and recognition of external landmarks. These methods depend on a prior map and proprioceptive sensors such as inertial measurement units (IMUs) and/or wheel encoders which are naturally immune to aforementioned adversary environmental conditions that may hinder exteroceptive sensors. PL is intended to be a low-cost and fallback solution when everything else fails.

We first propose a method named as proprioceptive localization assisted by magnetoreception (PLAM). PLAM employs a gyroscope and a compass to sense heading changes and matches the heading sequence with a pre-processed heading graph to localize the robot. Not all cases can be successful because degenerated maps may consist of rectangular grid-like streets and the robot may travel in a loop. To analyze these, we use information entropy to model map characteristics and perform both simulation and experiments to find out typical heading and information entropy requirements for localization.

We further propose a method which allows continuous localization and is less limited by map degeneracy. Assisted by magnetoreception, we use IMUs and wheel encoders to estimate vehicle trajectory which is used to query a prior known map to obtain location. We named the proposed method as graph-based proprioceptive localization (GBPL).

As a robot travels, we extract a sequence of heading-length values for straight segments from the trajectory and match the sequence with a pre-processed heading-length graph (HLG) abstracted from the prior known map to localize the robot under a graph-matching approach. Using HLG information, our location alignment and verification module compensates for trajectory drift, wheel slip, or tire inflation level.

With the development of communication technology, it becomes possible to leverage vehicle-to-vehicle (V2V) communication to develop a multiple vehicle/robot collaborative localization scheme. Named as collaborative graph-based proprioceptive localization (C-GBPL), we extract heading-length sequence from the trajectory as features. When rendezvousing with other vehicles, the ego vehicle aggregates the features from others and forms a merged query graph. We match the query graph with the HLG to localize the vehicle under a graph-to-graph matching approach. The C-GBPL algorithm significantly outperforms its single-vehicle counterpart in localization speed and robustness to trajectory and map degeneracy.

Besides, we propose a PL method with WiFi in the indoor environment targeted at handling inconsistent access points (APs). We develop a windowed majority voting and statistical hypothesis testing-based approach to remove APs with large displacements between reference and query data sets. We refine the localization by applying maximum likelihood estimation method to the closed-form posterior location distribution over the filtered signal strength and AP sets in the time window. Our method achieves a mean localization error of less than 3.7 meters even when 70% of APs are inconsistent.

To my parents, husband and sons

ACKNOWLEDGEMENTS

First and foremost I am extremely grateful to my advisor, Dr. Dezhen Song for his unwavering guidance, valuable advice, and relentless support throughout this research. I would also like to express the appreciation to my committee members: Dr. Suman Chakravorty, Dr. Dylan Shell, and Dr. Radu Stoleru, because of their insightful suggestions, input, and feedback to the knowledge and experiments of this study. Also, I would like to thank Dr. Jingang Yi for being my great research collaborator, and thanks to Y. Lu, J. Lee, C. Chou, B. Li, S. Yeh, A. Kingery, D. Wang, and S. Xie for their inputs and contributions to the NetBot Laboratory and my research projects. Setup of the experiment apparatus and collection of data could not have been possible without the assistance of J. Meng and A. Angert. I am especially grateful for the unending love, nurturing, and encouragement of all my family members. Last but not least, special thanks to everyone who has been supporting and praying with/for me during this time.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Professors Dezhen Song, Dylan Shell, and Radu Stoleru of the Department of Computer Science & Engineering and Professor Suman Chakravorty of the Department of Aerospace Engineering.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a fellowship from Texas A&M University and was supported in part by the National Science Foundation under IIS-1318638, NRI- 1426752, NRI-1526200, and NRI-1748161.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Nowadays mobile robots and autonomous vehicles (AVs) are changing our daily life in broad and dynamic ways. Mobile robots have the capability to move around in the environment with applications such as exploration, surveillance, manufacturing, transportation, agriculture, defense, search and rescue, domestic service, space, entertainment, education, security, etc. For example, ground robots search in collapsed buildings and marine vehicles are used to inspect bridge substructure. The fast-evolving AV technology is capable of sensing the environment and traveling with little or no human input and has the potential to drastically change modern transportation.

Navigation is a fundamental functionality for mobile robots and AVs. The main building blocks of navigation involve multiple tasks such as perception, localization, mapping, path planning, obstacle avoidance, trajectory following, dynamics and control. Perception tasks help in recognizing the environment by getting information from different sensors. These include road or traffic signals road extraction using computer vision analysis or road surface detection using lidar scans. Localization and mapping answers the basic questions: "where am I?" and "what is the world like?", correspondingly. Path planning is to find a path for a robot to move from source to destination. To navigate safely in the environment, obstacle avoidance keeps a robot from collision with detected objects. Trajectory following or lane keeping is the control of the robot to reach and follow a time parameterized reference geometric path. Robot dynamics is the relation between forces and moments acting on the robot and robot motion. Robot control system is the combination of hardware and software to program and control robots.

Localization is especially a critical navigation function for mobile robots. Knowing the robot's location is a prerequisite which helps making decisions for future movements.

In the past decades, localization approaches using different sensors have been explored which can be classified into two categories based on sensor modalities: exteroceptive sensors and proprioceptive sensors. Exteroceptive sensors mainly rely on the perception and recognition of landmarks in the environment to estimate location. Mainstream exteroceptive sensors include sonars, cameras, lidars and global position system (GPS) receivers. On the other hand, proprioceptive sensors, such as inertial measurement units (IMUs) [4] and wheel encoders, are inherently immune to external conditions.

In urban areas, common localization methods employ a GPS, a lidar, or a camera which are exteroceptive sensors relying on the perception and recognition of landmarks in the environment. However, GPS signals may be unavailable because high-rise buildings may block GPS signals in urban areas. Poor weather and lighting conditions may challenge all exteroceptive sensors. For example, cameras are prevalent external sensors for localization due to being low-cost and lightweight. However visual cues suffer with environmental changes such as variant light conditions or weather conditions, as well as being computationally expensive. A lidar is an accurate range sensor for localization and mapping but it is expensive and limited by weather conditions.

Recent sensor fusion approaches that combine an exteroceptive sensor, such as a camera or a lidar, with a proprioceptive sensor such as an IMU, greatly improve system robustness and have become popular in applications [5]. However, the sensor fusion approaches still strongly depend on exteroceptive sensors and cannot handle the aforementioned extreme environmental conditions.

Therefore, there is a critical need to improve the robustness of localization in both urban and indoor environments. To meet these challenges, we investigate low-cost localization methods using proprioceptive sensors. Proprioceptive localization (PL) refers to a new class of robot egocentric localization methods that do not rely on the perception and recognition of external landmarks. These methods are naturally immune to bad weather,

poor lighting conditions, or other extreme environmental conditions that may hinder exteroceptive sensors such as cameras or lidars. Inspired by biological systems, we combine proprioceptive sensors, such as IMUs and wheel encoders, with magnetoreception, to develop a map-based localization method to address the problem.

In this research, we first develop a minimalist and robust PL method for a robot/vehicle traveling in an urban area. We only employ a gyroscope and a compass. To be specific, we pre-process the prior map into a heading graph (HG) which is a new data structure capturing heading changes from one segment of road to adjacent segments. During the traveling, we process gyroscope and compass readings to obtain a heading change sequence and match the heading sequence with a pre-processed heading graph to localize the robot. We track the sensory and map uncertainties and model them in the process to formulate a sequential Bayesian estimation problem framework. Not all cases can be successful because degenerated maps may consist of rectangular grid-like streets and the robot may travel in a loop. To analyze these, we use information entropy to model map characteristics and perform both simulation and experiments to find out typical heading and information entropy requirements for localization. We have implemented our algorithm and tested it with both simulated and physical experiments. The results have confirmed our approach can localize vehicles on the map for non-degenerated cases.

However, localization is intermittent using heading sequence only. This motivates us to design a new method that enables continuous localization by considering wheel encoder inputs and is less limited by map degeneracy (e.g. rectilinear environments). In a nutshell, our method employs the proprioceptive sensors: IMUs, wheel encoders and assisted by magnetoreception to estimate vehicle trajectory and match it with a prior known map. This is non-trivial because 1) there is a significant drift issue in the dead reckoning process and 2) the true vehicle trajectory does not necessarily match the street GPS waypoints on the map due to the fact that a street may contain multiple lanes and street GPS waypoints may

be inaccurate. This determines that a simple trajectory matching would not work. Instead, we focus on matching features which are straight segments of the trajectory. We keep track of connectivity, heading and length of each segment which converts the trajectory to a discrete and connected query sequence. This allows us to formulate the problem as a probabilistic graph matching problem. As a robot/vehicle travels, we extract a sequence of heading-length values for straight segments from the trajectory and match the sequence with a pre-processed heading-length graph abstracted from the prior known map to localize the robot under a Bayesian approach. As the robot travels, we perform sequential Bayesian estimation until it converges to a unique solution. Also, we bound error drift in location alignment and verification after graph matching. We have implemented our algorithm and tested it in both simulated and physical experiments. The algorithm runs successfully and achieves localization accurate at the level that the prior map allows (less than 10m).

The previous two approaches still suffer from strong dependence on trajectory types and slow convergence. On the other hand, modern vehicle-to-vehicle (V2V) communication allows real time information exchange between vehicles. Combining the PL method with V2V communication, we design a new collaborative graph-based PL method. We extract trajectory features which are straight segments of trajectories and generate a merged query graph by combining inputs from neighboring vehicles. To facilitate the matching, we also pre-process the prior map which extends our heading-length graph (HLG) in [2] by adding supper-vertices based on three different vehicle rendezvous types. The localization problem becomes a graph-to-graph matching problem. Our algorithm outputs potential vehicle locations based on the maximization of belief functions which often quickly converges to actual location over time. We have implemented the algorithm and tested it against the existing approach [2]. Our algorithm significantly outperforms its single-vehicle counterpart in localization speed and is less sensitive to trajectory and map limitations.

Besides, we propose a PL method with WiFi in the indoor environment. This approach is designed to handle the inconsistent WiFi environments using proprioceptive sensors. Building on the existing WiFi fingerprinting approach, our method also employs Gaussian processes (GPs) to establish belief functions from prior collected WiFi reference data. Moreover, our approach utilizes two important designs to handle inconsistent access points (APs). First, majority voting is introduced to the initial matching phase which allows us to develop a statistical hypothesis test to filter out inconsistent APs that are obvious out of places. Second, we use a windowed approach by employing a window of recent RSS readings along with relative motion information provided by IMUs to develop posterior distribution of location. We formally derive the conditional distribution and determine the length of the time window by minimizing Shannon entropy. At last, we apply the maximum likelihood estimation (MLE) method to obtain refined localization results. We have implemented our algorithm and compared it with the state of the art k-Nearest-Neighbor (k-NN) approach. The experimental results show that our method outperforms its counterpart in inconsistent WiFi environments. Specifically, our algorithm achieves a mean localization error of less than 3.7 meters when 70% of APs are inconsistent.

The rest of this research is organized as follows. We begin with a review of literature in Chapter 2. In Chapter 3, we present the proprioceptive localization assisted by magnetoreception (PLAM) which is a minimalist intermittent heading-based approach. Chapter 4 presents Graph-based proprioceptive localization (GBPL) using a Bayesian piece-wise trajectory matching approach. Chapter 5 presents our V2V collaborative graph-based proprioceptive localization (C-GBPL). Chapter 6 we present the indoor localization in inconsistent WiFi environment. Chapter 7 concludes the dissertation and discusses future work directions.

2. RELATED WORK

Our work is related to mobile robot navigation, simultaneous localization and mapping (SLAM), dead-reckoning, and map representation.

2.1 Mobile Robot Navigation

Robot navigation is a fundamental problem in robotics research. The main building blocks of navigation involves multiple tasks such as perception, localization, mapping, path planning, obstacle avoidance, trajectory following, dynamics and control. Perception tasks help in recognizing environment by getting information from different sensors [6, 7]. These include road or traffic signals road extraction using computer vision analysis or road surface detection using lidar scans. Localization and mapping answers the basic questions: "where am I?" and "what is the world like?", correspondingly. Path planning is to finds a path for a robot to move from the source to destination. To navigate safely in the environment, obstacle avoidance keeps a robot from collision with detected objects. Ultrasonic [8] or lidars [9] are commonly used to detect obstacles. Trajectory following or lane keeping [10] is the control of robot to reach and follow a time parameterized reference geometric path. Robot dynamics and control [11] consider the problem that how to execute a given joint trajectory on a robot. Robot dynamics is the relation between forces and moments acting on the robot and robot motion. Robot control system is the combination of hardware and software to program and control robots.

2.2 SLAM

In dealing with the robot navigation in unknown environments, localization and mapping are usually performed simultaneously [12]. The architecture of recent SLAM systems mainly consist of a front-end [13–15] and back-end [5, 14, 16–18] running in parallel for

the sake of efficiency. The front-end performs feature extraction and tracking from sensory data. The back-end focuses on state estimation, refines the trajectory/map and closes loops occasionally.

To estimate robot states, there are two popular computation frameworks for state estimation: filtering approaches (e.g., [5, 16]) and optimization approaches (e.g., [14, 17, 18]). Filtering approach is to estimate robot state with its previous state, and update it with observations. The filtering approach maintains compact state variables and correlation history but the accuracy is not very high. Optimization approach is to take the robot states in multiple frames as parameters and optimize them over the observations from those frames. The accuracy of the optimization approach is good, but the computation cost is high. In this study, we use filtering approaches to estimate robot trajectories and optimization approaches to localize the robots.

Under different applications, odometry and localization are popular related topics of SLAM. When loop closure is not considered in the problem, SLAM problem is reduced to an odometry estimation problem. In some robotics applications, a prior map is given and only localization problems are considered. Since proprioceptive sensors cannot recognize landmarks in the environment, a map is needed for localization. In this work, we focus on localization which is the problem of estimating robot poses (position and orientation) with respect to a known map which also has its own advantage in computational cost.

2.2.1 Different Sensor Modalities

To perform SLAM, a robot is equipped with on-board sensors to receive sensory data. SLAM can be performed with different sensor modalities such as camera [14, 19–23], lidar [15, 24–26], global positioning system (GPS) [27–29], and inertial measurement unit (IMU) [4, 30]. We can classify the methods into two categories based on sensor modalities: exteroceptive sensors and proprioceptive sensors.

Exteroceptive sensors. Exteroceptive sensors mainly rely on the perception and recognition of landmarks in the environment to estimate location. Mainstream exteroceptive sensors include cameras [14, 19, 20] and laser range finders [24–26]. Cameras are prevalent external sensors for localization due to being low-cost and lightweight, however visual cues suffer with environmental changes such as variant light conditions or dynamic environments, as well as being computationally expensive. Lidar is an accurate range sensor for localization and mapping but it is expensive and limited by specific environments (e.g. heavy rains). GPS receiver [27, 29] is another commonly-used sensor but it malfunctions when the vehicle travels close to high-rise buildings or inside tunnels. Additionally, it also suffers from large power consumption for mobile devices. Localization can be performed with different external sensors or their combinations such as a camera, a lidar, or a GPS, but sill cannot handle the aforementioned challenging conditions.

Proprioceptive sensors. On the other hand, proprioceptive sensors, such as IMUs [4] and wheel encoders [31], are inherently immune to external conditions. IMUs in mobile devices have recently drawn attention from researchers because of its availability and energy-efficiency for localization [30]. However, they are more susceptible to error drift and suffer from limited accuracy.

Recent sensor fusion approaches that combine an exteroceptive sensor, such as a camera or a laser ranger finder, with a proprioceptive sensor such as an IMU, greatly improve system robustness and become popular in applications [5]. For example, visual-inertial sensor fusion (VINS)([13, 32]) methods are applied to a variety of domains such as autonomous vehicles, virtual and augmented reality, and flying robots. However, the sensor fusion approaches still strongly depend on exteroceptive sensor and cannot handle the aforementioned challenging conditions. In this research, we use proprioceptive sensors which are immune to the external weather conditions.

2.3 Dead Reckoning

To utilize proprioceptive sensors for navigation, dead reckoning integrates sensor measurements to compute robot/vehicle trajectory. The sensor measurements often include readings from accelerometers, gyroscopes, and/or wheel encoders [33]. There are many applications using the dead reckoning approach such as autonomous underwater vehicles (AUVs) [34] and pedestrian step measurement [30, 35]. To estimate the state of the robot/vehicle, filtering-based schemes such as unscented Kalman filter (UKF) [36] and particle filter (PF) [37, 38] are frequently employed. However, the nature of dead reckoning causes it to inevitably accumulate errors over time and lead to significant drift. To reduce the error drift, different methods have been proposed such as applying velocity constraint on wheeled robots [39] and modeling the wheel slip for skid-steered mobile robots [33]. These approaches have reduced error drift but cannot remove it completely. Error still accumulates over time and causes localization failure. To fix the issue, we will show that drift can be bounded to map accuracy level by using map matching if the filtering-based approach with graph matching are combined.

2.4 World Modeling-Map Representation

A map is the environment representation. Since proprioceptive sensors cannot recognize landmark in the environment, a map is needed for localization. Moreover, to improve the accuracy of localization, using a prior map is a solid approach because it can alleviate accumulated drift in localization from time to time by comparing with the prior map [19, 30, 40–43]. According to [12], map representation can be classified into two categories: the location-based and the feature-based.

Location-based. The location-based maps are represented with specific locations of objects. For example, those existing geographic maps consisted of coordinate of locations such as OpenStreetMaps™ (OSM) [44] and Google Maps [45]. Geographic maps have

been widely used to improve upon GPS measurements and there are common measures being used such as point-to-point, point-to-curve, curve-to-curve matching or advanced techniques [46]. For example, occupancy grid maps is to partition the open space into equalized grid cells and then assign a probability of occupation to each cell [47].

Feature-based. In feature-based maps, each object is a feature of interest with its location, and it only describes the environment with those features. To localize a vehicle on a road map, feature-based maps are a better fit because they are more efficient to localize vehicles on specific roads rather than the whole map space. Road maps belong to feature-based maps since they consist of the attributes (e.g. name, orientation) of the roads as features and the topological information among all features. The typical way of describing road networks as a topological framework is to define road segments as nodes and road intersections as edges. The definition is applicable to map odometry data with both position and orientation information. Recently, online road maps have become more and more popular due to their availability and convenience, e.g. OpenStreeMaps™ (OSM) [44] and Google Maps [45]. Another example is ORB features [48] for visual simultaneous localization and mapping. In this work, we extract heading-length graph from geographic maps which converts a location-based map to a feature-based map to facilitate robust localization which also reduces graph size to speed up computation in the process.

3. PROPRIOCEPTIVE LOCALIZATION ASSISTED BY MAGNETORECEPTION: A MINIMALIST INTERMITTENT HEADING-BASED APPROACH*

We are interested in developing a minimalist and robust localization method for a robot/vehicle traveling in urban area. We want our localization method to be immune to environment challenges such as bad weather and poor illumination conditions. Imagining a vehicle driving in a stormy or foggy night, no other sensors would work. As a fall back solution, we do not want to rely on the perception and recognition of landmarks from exteroceptive sensors such as a lidar or a camera. We consider it as a low-cost alternative to existing localization methods and the global positioning system (GPS), because GPS signals may be obstructed by high-rise buildings in urban area. We only employ a gyroscope and a compass. We name our localization method proprioceptive localization assisted by magnetoreception (PLAM).

Fig. 4.1 illustrates PLAM method which requires a prior urban map. We pre-process the prior map into a heading graph (HG) which is a new data structure capturing heading changes from one segment of road to adjacent segments. During the traveling, we process gyroscope and compass readings to obtain a heading change sequence (see Fig. 3.1(b)). At the beginning, we often have many candidate solutions (see Fig. 3.1(c)). We track the sensory and map uncertainties and model them in the process to formulate a sequential Bayesian estimation problem framework for PLAM. As the vehicle travels, the heading sequence grows, the number of candidates reduces and the Bayesian probability distribution converges to one solution (see Fig. 3.1(d)) for most cases. Not all cases can be successful because there exist degenerated maps consisting of rectangular grid streets and the vehicle

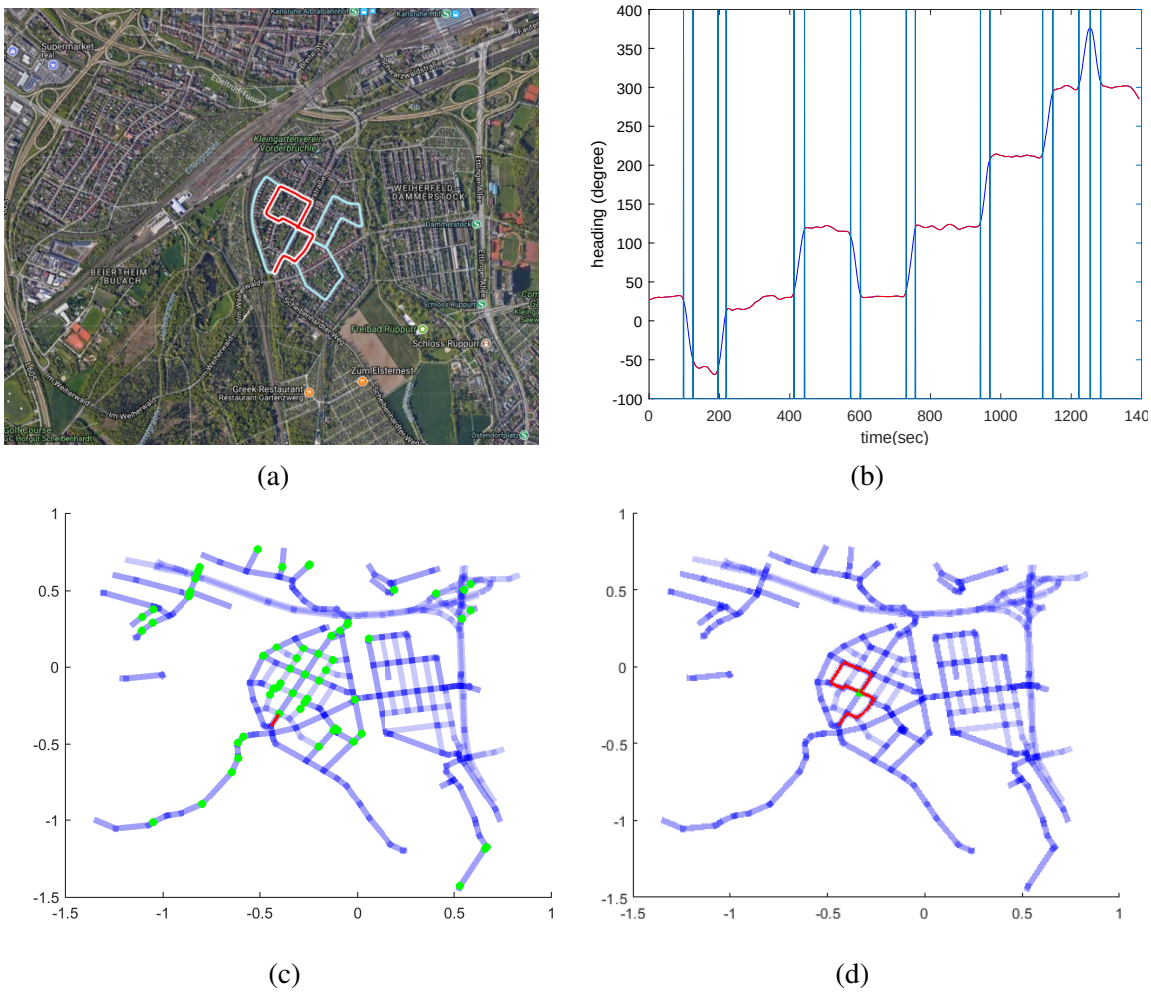


Figure 3.1: An illustration of PLAM method: inputs include (a) a prior urban map and (b) heading changes computed from the gyroscope and the compass on-board the vehicle. The candidate locations (in green dot shown in (c)) gradually converge to one location shown in (d) as more observations of heading changes arrive. Reprinted with permission from [1] © 2019 IEEE.

may travel in a loop. To analyze these, we employ information entropy to model map characteristics and perform both simulation and physical experiments to identify typical

heading and information entropy requirements for localization.

We have implemented the PLAM algorithm and tested it using both our own collected data and an open dataset. The algorithm runs successfully and the information entropy analysis results are consistent between simulation and physical experiments. The results have confirmed that PLAM can localize vehicles on the map for non-degenerated cases.

3.1 Related Work

Most localization methods employs exteroceptive sensors such as a camera [14,19,20], lidar [24–26], and GPS receiver [27,29]. Exteroceptive sensors are susceptible to environmental changes and signal variations and are often used in combination with proprioceptive sensors such as inertial measurement units (IMU) [4, 30] or wheel encoders [31]. Mostly a proprioceptive method, PLAM attempts to provide a fall back localization solution when everything else fails due to bad weather or poor visibility conditions. Our method uses the magnetometer and gyroscopic readings which are subject to bias and calibration issues. Sensor bias estimation and calibration is a well-studied area [49, 50]. In this work, we assume sensors have been calibrated before usage.

Due to its reliance on angle measurements, PLAM is related to bearing-based localization. Bearing-based localization relies on observing the bearing towards landmarks to localize the client. Many rely on fixed landmarks with known locations. The robot localizes itself by measuring the bearings to a sufficient number of landmarks through triangulation [51–53]. Most bearing-based localizations are still exteroceptive sensing. Our method employs the headings of the robot without observing or recognizing landmarks.

PLAM needs a prior map to provide reference for matching with a heading sequence. Map-based localization is a common approach in robotics [30,40–43]. A map of the environment is a representation of objects and locations. The representation can be classified into two types: the location-based and the feature-based [12]. The location-based maps

consist of a list of objects, where each object corresponds to a specific location. Existing geographic maps such as OpenStreetMaps™ (OSM) [44] and Google Maps [45] are indexed by longitude and latitude can be viewed as location-based maps. In the feature-based maps, each object is a feature of interest with its location. Visual features such as ORB [48] are often used for constructing the feature-based maps for visual simultaneous localization and mapping. Our approach extracts heading graphs out of geographic maps and actually converts a location-based map to a feature-based map for localization purposes.

Closely-related works include [43, 54] which focus on position correction with orientation changes but they still depend on camera inputs for motion estimation. As an independent work, Funke et al. [55] propose a localization algorithm using electronic compass readings, which is represented as path shapes for matching. They introduce a special data structure to store all possible shortest paths in the map to transfer the map matching problem to a pattern search in texts. The data structure requires a large space storage and has the limitation that not all possible paths can be found (only shortest paths). Moreover, the data structure only allows pattern search if the impression of electronic compass readings is within a fixed range. Their approach relies on global trajectory matching and assumes constant velocity while our method can be viewed as feature-based matching without velocity constraints. Moreover, compass alone is not very reliable due to frequent electromagnetic interference. We combine a gyroscope with a digital compass and monitor their uncertainties in the localization process, which improves robustness.

3.2 Problem Formulation

A robot or vehicle equipped with a gyroscope and a digital compass navigates in an area with a given prior map, e.g. OSM. We have the following assumptions:

a.1 The robot only performs forward motion during the localization process.

a.2 The gyroscope and the compass are co-located, pre-calibrated, and fixed inside the robot.

As part of the input of the problem, a prior road map consisting of a set of roads with GPS way points are required. The typical distance between adjacent way points is around 20m. Gyroscope and compass readings are time-stamped and synchronized. We denote the sampling intervals of gyroscope and compass readings by τ_ω and τ_ϕ, respectively. A gyroscope often has a higher sampling frequency than that of a compass and thus we have $\tau_\phi = c_\omega \tau_\omega$ where $c_\omega \geq 1$. Common notations are defined as follows,

- $\mathcal{M}_p := \{\mathbf{x}_m = [x_m, y_m]^\mathsf{T} | m \in \mathscr{M}\} \subset \mathbb{R}^2$ represents the prior road map which is a set of GPS positions where $\mathscr{M}$ is the position index set.
- $\mathcal{R} = \{\mathcal{R}_{i_r} | i_r \in \mathcal{I}_r\}$ represent roads on $\mathcal{M}_p$ where $\mathcal{I}_r$ is the road index set. Each $\mathcal{R}_{i_r} := \{\mathbf{x}_m = [x_m, y_m]^\mathsf{T} | m \in \mathscr{R}(i_r)\}$ is a set of ordered GPS positions where $\mathscr{R}(i_r)$ collects all indexes of positions on $\mathcal{R}_{i_r}$. $\mathcal{R}_{i_r} \subset \mathcal{M}_p$.
- $\omega_{0:t} \in \mathbb{R}^3$ denotes gyroscope angular velocities readings up to time t.
- $\phi_{0:t_\phi} \in \mathbb{R}$ denotes electronic compass readings up to time t. Note that $\omega_{0:t}$ contains more entries than $\phi_{0:t_\phi}$ due to the higher sampling frequency.

The PLAM problem is defined as follows.

Problem 1. *Given $\mathcal{M}_p$, $\mathcal{R}$, $\omega_{0:t}$ and $\phi_{0:t_\phi}$, localize the robot after its heading changes.*

It is worth noting that we can localize the robot only when the robot changes its heading. This is an intermittent localization. Also, the localization accuracy is determined by map accuracy.

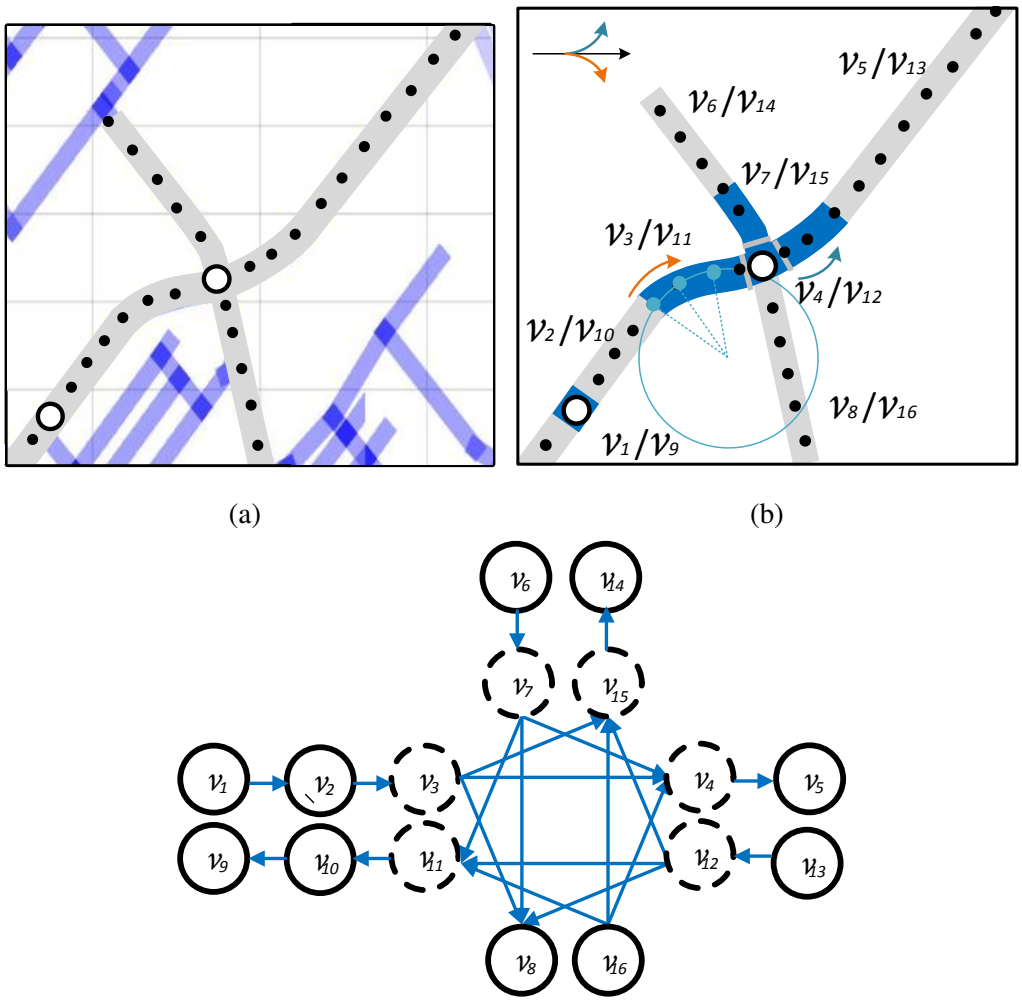


Figure 3.2: [HG construction. (a) A prior road map: two bidirectional roads with 2D GPS coordinates represented by dots and road connectivities shown in circles. (b) Overlay of edges and vertices for gray roads in (a). We use $\cdot/\cdot$ to represent a bi-directional road. Vertices are shown in gray and edges are in blue. (c) Constructed HG, where vertices with dotted lines have zero length. (Best viewed in color.) Reprinted with permission from [1] © 2019 IEEE.

3.3 PLAM Modeling and Design

Our algorithm contains three main blocks: 1) Heading graph (HG) construction where we estimate road curvature changes to capture orientation change and construct an HG. 2)

Query sequence generation, where we estimate heading orientation using gyroscope and compass readings to capture orientation change to generate a query sequence. 3) Location inference, where we match the query sequence with the HG to find the current location of the vehicle. We begin with HG construction.

3.3.1 HG Construction

The prior road map $\mathcal{M}_p$ cannot be directly used for localization purposes. We pre-process $\mathcal{M}_p$ to construct an HG to facilitate heading matchings. We use Fig. 3.2 as an example. Fig. 3.2(a) is the blowup view of the prior road map showing GPS 2D positions in black dots and road intersections in circles. We denote the set of road intersections by $\mathcal{I}_{\mathbf{x}}$. We denote the HG by $\mathcal{M}_h = \{\mathcal{V}_h, \mathcal{E}_h\}$ where $\mathcal{V}_h$ is the vertex set and $\mathcal{E}_h$ and is the edge set. A vertex $v_i \in \mathcal{V}_h$ represents a straight and continuous road segment with neither orientation changes nor intersections. An edge $e_{i,i'} \in \mathcal{E}_h$ characterizes the orientation change between the connected two vertices v_i and $v_{i'}$. $\mathcal{M}_h$ is a directed graph (see Fig. 3.2(c)). $\mathcal{M}_h$ have two types of edges: road intersections and curve segments; and two types of vertices: long straight segment vertices and short transitional segment vertices. The long straight segment vertices are used for heading matching later. The short transitional segment vertices are often formed between curve segments or curved roads entering intersections. See Fig. 3.2(b) for examples.

3.3.1.1 Road Segmentation Based on Curvature Estimation

To build $\mathcal{M}_h$, we first split each road $\mathcal{R}_{i_r}$ at road intersections and then further segment it into two types of segments to capture orientation changes: straight segments and curved segments. Fig. 3.2(b) shows an example.

We consider a complex road can be approximated with straight and curve segments by jointing short segments together. To find straight segments and curve segments, we identify the locations when the road curvature sign changes. Thus we approximate the

road curve by a circular arc where its curvature is defined to be the reciprocal of radius as illustrated in Fig. 3.2(b). For each $\mathbf{x}_m \in \mathcal{R}_{i_r}$, the set of its neighboring points are $\mathcal{N}_{\mathbf{x}_m} = \{\mathbf{x}_{m'} = [x_{m'}, y_{m'}]^{\mathsf{T}} | m-2 \leq m' \leq m+2\}$, where $\mathbf{x}_{m'} \in \mathcal{R}_{i_r}$. For $\mathbf{x}_m$ with insufficient neighboring points, we take it as straight segment and thus assign zero curvature. We estimate the circle using all $\mathbf{x}_{m'} \in \mathcal{N}_{\mathbf{x}_m}$. The circle equation is written by $(x_{m'} - x_{c,m})^2 + (y_{m'} - y_{c,m})^2 = r_m^2$ with center at $\mathbf{x}_{c,m} = [x_{c,m}, y_{c,m}]^{\mathsf{T}}$ and radius r_m. We represent the circle equation using matrix form by

$$\tilde{\mathbf{x}}_{m'}^{\mathsf{T}} \mathbf{Q}_m \tilde{\mathbf{x}}_{m'} = 0, \forall \mathbf{x}_{m'} \in \mathcal{N}_{\mathbf{x}_m}, \tag{3.1}$$

where $\tilde{\mathbf{x}}_{m'} = [x_{m'}, y_{m'}, 1]^{\mathsf{T}}$ and $\mathbf{Q}_m = \begin{bmatrix} 1 & 0 & -x_{c,m} \\ 0 & 1 & -y_{c,m} \\ -x_{c,m} & -y_{c,m} & r_m^2 \end{bmatrix}$. By using all measurements $\mathbf{x}_{m'}$, we estimate $\mathbf{x}_{c,m}$ and r_m by minimizing

$$[\mathbf{x}_{c,m}^{\mathsf{T}}, r_m] = \underset{\mathbf{x}_{c,m}, r_m}{\arg\min} \sum_{\mathbf{x}_{m'} \in \mathbf{X}_m} \tilde{\mathbf{x}}_{m'}^{\mathsf{T}} \mathbf{Q}_m \tilde{\mathbf{x}}_{m'}. \tag{3.2}$$

The curvature of a circle is $\frac{1}{r_m}$. Straight lines can be considered as the degenerate case of circles with infinite radius. To avoid ill-condition for solving (3.2), we first check if the points fit a line well. If not, we apply (3.2). To segment a road by the sign of road curvature, we define the label function $l(m)$ for $\mathbf{x}_m$:

$$l(m) = \begin{cases} 0, & \frac{1}{r_m} \leq \tau_\kappa \text{ (on a straight segment)}, \\ 1, & \frac{1}{r_m} > \tau_\kappa \text{ and } \langle \mathbf{x}_m - \mathbf{x}_{c,m}, \mathbf{n} \rangle > 0 \text{ (positive curvature)}, \\ -1, & \text{otherwise (negative curvature)}, \end{cases} \tag{3.3}$$

where τ_κ is a thresholding parameter used to classify curve segment and straight segments, the symbol $\langle \cdot, \cdot \rangle$ represents the inner product between vectors, and $\mathbf{n} = [0, 1]^\mathtt{T}$ is the unit vector of the geographic north direction.

After assigning the label value using $l(m)$ for all $\mathbf{x}_m \in \mathcal{R}_{i_r}$, the segmentation of $\mathcal{R}_{i_r}$ takes place at road intersections ($\mathbf{x}_m \in \mathcal{I}_\mathbf{x}$) and positions where $l(m) \neq l(m-1)$. We denote the segments of $\mathcal{R}_{i_r}$ by $\{\mathcal{R}_{i_r,s} | s \in \mathcal{S}(i_r)\}$ where $\mathcal{S}(i_r)$ is the index set of segmented $\mathcal{R}_{i_r}$.

3.3.1.2 HG Vertex Orientation Estimation

With all roads segmented, we compute orientation for vertices corresponding to those long straight road segments. Each vertex contains the following information $v_i = \{\theta_i, \mathbf{x}_{i,s}, \mathbf{x}_{i,e}\}$, where orientation $\theta_i \in (-\pi, \pi]$ is the angle between the geographic north and the orientation of the road segment, $\mathbf{x}_{i,s} = [x_{i,s}, y_{i,s}]^\mathtt{T}$ is the starting position, and $\mathbf{x}_{i,e} = [x_{i,e}, y_{i,e}]^\mathtt{T}$ is the ending position of the road segment. Note that θ_i also depends on vehicle traveling direction and hence $\mathcal{M}_h$ is a directed graph. We only perform orientation estimation if $\|\mathbf{x}_{i,s} - \mathbf{x}_{i,e}\| > t_l$ where t_l is the threshold for road segment length. Only long road segments will be used in localization which defines vertex subset $\mathcal{V}_{h,l} \subset \mathcal{V}_h$ corresponding to long straight segments.

We use all 2D positions on the segment to form $\mathbf{X}_i = [\mathbf{x}_{i,s}^\mathtt{T}, \cdots, \mathbf{x}_{i,e}^\mathtt{T}]^\mathtt{T}$ and compute θ_i. Given $\mathbf{X}_i$, we fit the positions using the linear model $y = a_1 x + a_0$ where a_1 and a_0 are the parameters. Using least squares estimation, the estimated parameter is $h(\mathbf{X}_i) = [a_1, a_0]^\mathtt{T} = (\mathbf{A}^\mathtt{T}\mathbf{A})^{-1}\mathbf{A}^\mathtt{T}\mathbf{b}$, where $\mathbf{A} = \begin{bmatrix} 1 & , \cdots , & 1 \\ x_{i,s} & , \cdots , & x_{i,e} \end{bmatrix}^\mathtt{T}$ and $\mathbf{b} = [y_{i,s}, \cdots, y_{i,e}]^\mathtt{T}$. We denote the estimated $\mathbf{x}_{i,s}$ and $\mathbf{x}_{i,e}$ by $\hat{\mathbf{x}}_{i,s}$ and $\hat{\mathbf{x}}_{i,e}$, where $\hat{\mathbf{x}}_{i,s} = [x_{i,s}, a_1 x_{i,s} + a_0]^\mathtt{T}$ and

$\hat{\mathbf{x}}_{i,e} = [x_{i,e}, a_1 x_{i,e} + a_0]^\intercal$. We compute θ_i by

$$\theta_i = f_\theta(\mathbf{X}_i) = \text{atan2}\big(\langle \frac{\hat{\mathbf{x}}_{i,e} - \hat{\mathbf{x}}_{i,s}}{||\hat{\mathbf{x}}_{i,e} - \hat{\mathbf{x}}_{i,s}||}, \mathbf{n}\rangle\big). \tag{3.4}$$

Through derivation, we rewrite $\langle \frac{\hat{\mathbf{x}}_{i,e} - \hat{\mathbf{x}}_{i,s}}{||\hat{\mathbf{x}}_{i,e} - \hat{\mathbf{x}}_{i,s}||}, \mathbf{n}\rangle = g([a_1, a_0]^\intercal) = \frac{|a_1|}{a_1^2+1}$.

The GPS errors in each entry of $\mathbf{X}_i$ affects the accuracy of θ_i. We can derive the distribution of θ_i since GPS errors can be modeled as Gaussian distribution. We assume GPS measurement noise to be uncorrelated because GPS points from map do not necessarily coming from the same time series. From the first order approximation of error propagation [56], the variance of θ_i in (3.4) is $\sigma_{\theta_i}^2 = J_\theta \Sigma_\theta J_\theta^\intercal$, where $J_\theta = \frac{\partial f_\theta}{\partial \mathbf{X}_i} = \frac{\partial f_\theta}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial \mathbf{X}_i}$ and $\Sigma_\theta = \sigma_g^2 \mathbf{I}$ given the GPS measurement variance σ_g^2. According to [19], typical consumer grade navigation systems offer positional accuracy of around 10m. The distribution of θ_i that characterizes its uncertainty is

$$\theta_i \sim \mathcal{N}(\mu_{\theta_i}, \sigma_{\theta_i}^2). \tag{3.5}$$

3.3.2 Query Sequence Generation

Gyroscope readings $\omega_{0:t}$ and compass readings $\phi_{0:t_\phi}$ can help us extract orientation and construct a query heading sequence $\mathcal{Q}_t$. To improve the robustness, we only keep headings when the vehicle is traveling on long and straight road segments. This means the headings should be stable and constant in a long stretch of travel time. To obtain the query sequence: (1) we first utilize an EKF-based approach using $\omega_{0:t}$ and $\phi_{0:t_\phi}$ to estimate heading, (2) sequence segmentation to obtain stable orientations, and (3) remove headings that do not correspond to long and straight road segments.

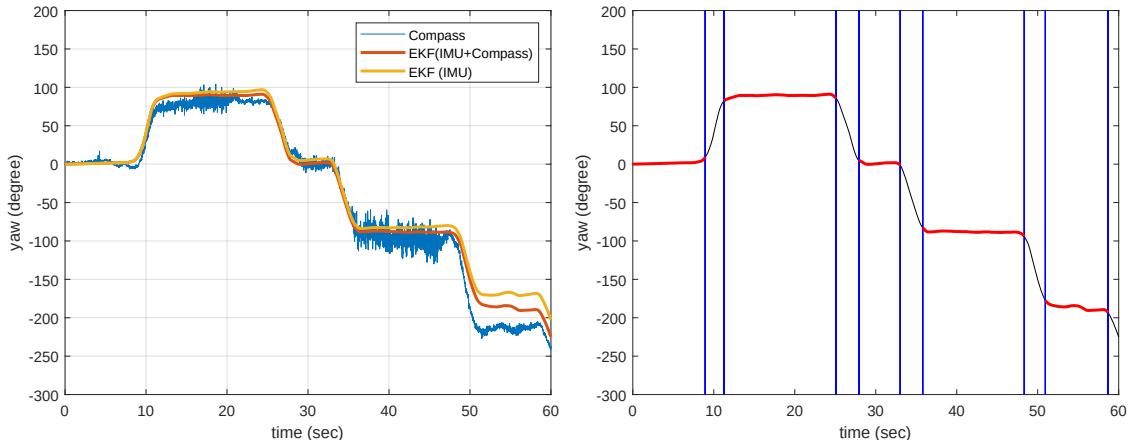


Figure 3.3: (a) Heading estimation by fusing gyroscope and compass readings. Blue line shows raw compass readings, orange line shows result using gyroscope only and red line shows result using gyroscope and with compass readings. (b) Query representations. Black line is estimated orientation using EKF, blue vertical lines are indices where gyroscope data are segmented, red lines mark out stable heading segments with no orientation changes. Reprinted with permission from [1] © 2019 IEEE.

3.3.2.1 Heading Orientation Estimation

Given $\omega_{0:t}$ and $\phi_{0:t_\phi}$, we employ an EKF-based approach to estimate heading orientation [57–59]. Body frame $\{B\}$ is gyroscope body frame, and the fixed inertial coordinate system $\{I\}$ shares origin with $\{B\}$ at the initial pose. Its Z axis is vertical and points upward. Frame $\{I\}$'s X and Y plane is a horizontal plane parallel to the ground plane with Y axis pointing to magnetic north direction. Both frames are right hand coordinate systems. $\mathbf{\Theta} := [\alpha, \beta, \gamma]^\mathsf{T}$ in X-Y-Z order representing roll, pitch, and yaw angles. In the state representation, we denote state vector by $\mathbf{\Theta}^I(t)$ being its discrete format and superscripts indicate in which frame the vector is defined. The transformation from $\{I\}$ to $\{B\}$ is the Z-Y-X ordered Euler angle rotation. The state transition equations for $\omega_{0:t}$ are described as follows:

$$\dot{\mathbf{\Theta}}^I = {}^I_B\mathbf{E}(\omega) + \mathbf{c}_\gamma, \tag{3.6}$$

where $\mathbf{c}_\gamma = [0\ 0\ \phi_0]^\mathrm{T}$ is the initial orientation determined by ϕ_0 and

$$ {}^{I}_{B}\mathbf{E} = \begin{bmatrix} 1 & \sin\alpha\tan\beta & \cos\alpha\tan\beta \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha/\cos\beta & \cos\alpha/\cos\beta \end{bmatrix} $$

is the rotation rate matrix from $\{B\}$ and $\{I\}$. We take $\phi_{0:t_\phi}$ as observations of γ values in EKF model when compass readings are available:

$$ \gamma(t) = \phi(t), \tag{3.7} $$

where $t = c_\omega t_\phi$.

Since compass readings may be The orientation of the device is represented by three Euler angles polluted by nearby electrical devices or materials, we can recognize faulty readings by cross-validating with IMU reading and not using them as EKF observations. From the coordinate definition, the heading is γ in $\{I\}$ and is denoted by $\gamma_{0:t}$. As an example, Fig. 3.3(a) shows $\phi_{0:t_\phi}$ in blue, estimated heading orientation result using only $\omega_{0:t}$ in orange, and $\gamma_{0:t}$ in red.

3.3.2.2 Stable Heading Sequence Generation

To capture the trend of orientation change of $\gamma_{0:t}$, we detect change points of $\gamma_{0:t}$ and segment data into a set of non-overlapping, consecutive segments. We employ the sliding window algorithm [60] to segment $\gamma_{0:t}$. The sliding window starts from the time at γ_0 which is the anchor point of potential segment and the segment is grown if the error of line-fitting data segments [61] does not exceed error bound. This filtering can help smooth out the situation when vehicles change lanes on a long and straight roads. Until some time t', the error is greater than the threshold, the subsequence from the anchor to $t' - 1$ is

transformed into a segment. The process repeats until the entire sequence is covered. To generate the query sequence, we extract segments with absolute slope smaller than 10^{-4} and obtain the segmentation indices for the sequence (e.g. blue lines in Fig. 3.3(b)). Red horizontal segments are candidate stable headings, which are obtained by apply the indices to the original $\gamma_{0:t}$ from EKF.

3.3.2.3 Remove False Positive Segments

We want to extract headings corresponding to the time when the vehicle is driving along a long and straight road segments. However, the stable headings may be caused by vehicle stopping at curved roads or slowing driving over a short distance which creates false positives. At the same time, we should allow the vehicle to stop or change speed while traveling along long and straight roads to accommodate different traffic conditions. We can filter false positive segments by observing the zero-crossing properties of angular speed. If the vehicle stops or slows down before resuming its speed, the beginning and the ending moments of the angular velocities are determined by the road curvature and its traveling speed. For curved road, the angular velocity starts from a non-zero value and reaches zero or a very small value during stopping or slow driving. If the road is straight, the angular velocity remains close to zero at all time. Of course, there may be occasional noises caused by vehicle suspension during stop and go but they can be filtered out using low pass filters. Hence we can remove false positive segments and obtain the set of query sequence by $\Theta_q = \{\Theta_{q,k} | k = 1, \cdots, n\}$ where k is query data index, n is the cardinality, and each subset $\Theta_{q,k}$ corresponding to continuous observations from EKF for a stable segment (e.g. a red segment in Fig. 3.3(b)).

3.3.3 Location Inference

Given Θ_q, we search for the best matching robot trajectory as a sequence of vertices in $\mathcal{M}_h$.

3.3.3.1 Sequence Matching

We start with sequence matching. Given $\Theta_q = \{\Theta_{q,k} | k = 1, \cdots, n\}$, let us denote the orientation of vertices in $\mathcal{M}_h$ by $\Theta_h = \{\theta_k | k = 1, \cdots, n\}$ correspondingly. We define $\overline{\theta_{q,k}}$ as the sample mean orientation of segment $\Theta_{q,k}$ which contains $n_{\theta_{q,k}}$ observations of random variable $\theta_{q,k}$. $\theta_{q,k}$ has a variance of $\sigma^2_{\theta_{q,k}}$ obtained from EKF. We denote the mean vector of Θ_q by $\mu_{\Theta_q} = [\mu_{\theta_{q,1}}, \cdots, \mu_{\theta_{q,n}}]$ and the mean vector of Θ_h by $\mu_{\Theta_h} = [\mu_{\theta_1}, \cdots, \mu_{\theta_n}]$. Due to independence in sensor noises and the mean difference between two normal distributions follows Student's t-distribution, the matching probability between Θ_q and Θ_h is

$$P(\mu_{\Theta_q} = \mu_{\Theta_h} | \Theta_q, \Theta_h) = \prod_{k=1}^{n} P(\mu_{\theta_{q,k}} = \mu_{\theta_k} | \Theta_{q,k}, \theta_k) \tag{3.8}$$

$$\propto \prod_{k=1}^{n} f_T(t(\theta_k, \theta_{q,k})), \tag{3.9}$$

where $f_T(t(\theta_k, \theta_{q,k}))$ is the probability density function of Student's t-distribution, and

$$t(\theta_k, \theta_{q,k}) = \frac{\theta_k - \overline{\theta_{q,k}}}{\sqrt{\sigma^2_{\theta_k} + \sigma^2_{\theta_{q,k}} / n_{\theta_{q,k}}}}. \tag{3.10}$$

According to (5.2), sequence matching is considered as multiple pair matching. For each pair $(\theta_k, \theta_{q,k})$, it is a hypothesis testing with

$$H_0 : \mu_{\theta_{q,k}} = \mu_{\theta_k} \tag{3.11}$$

$$H_1 : \text{otherwise} \tag{3.12}$$

Hypothesis H_0 can be tested with significance level $1 - \alpha$ where α is a small probability. Since this is a two-tailed distribution, we choose $t_{\alpha/2,\nu}$ as the t statistic that has a cumula-

tive probability of $(1 - \frac{\alpha}{2})$ where ν is the degree of freedom. That is, $P(t > t_{\alpha/2}) = \alpha/2$. Thus we reject H_0 if

$$|t(\theta_k, \theta_{q,k})| > t_{\alpha/2,\nu}. \tag{3.13}$$

This hypothesis test serves as a building block of the localization scheme. By sequentially applying the hypothesis on each corresponding pair $(\mu_{\theta_{q,k}}, \mu_{\theta_k})$ from Θ_q and Θ_h, we can determine whether Θ_h represents the actual trajectory and hences the vehicle location is obtained.

3.3.3.2 Localization Scheme and Analysis

The remaining problem is whether this sequence of hypothesis would converge to the true trajectory. Let us define binary event $C_k = 1$ if the node k in Θ_h is the actual location, and binary event $B_k = 1$ if $\mu_{\theta_{q,k}} = \mu_{\theta_k}$ is true. With these definitions, candidate set Θ_h represents the true trajectory is the joint event of $C_1 C_2 \cdots C_n = 1$. Let $n_v = |\mathcal{V}_{h,l}|$ be the cardinality of $\mathcal{V}_{h,l}$ and n_b be the expected number of neighbors for each vertex. However, we do not know $C_1 C_2 \cdots C_n$. We only know the joint event $B_1 B_2 \cdots B_n$ by performing the sequence matching.

To facilitate our analysis, we assume there are k_d levels of distinguishable discrete headings in the $[0, 2\pi)$ and each vertex heading takes a heading value with an equal probability of $1/k_d$. This is a very rudimentary way to describe map/trajectory property. We know $n_v \gg k_d \geq n_b$ is generally true for most maps. We have the following lemma.

Lemma 1. *The conditional probability that Θ_h is the true matching sequence given that Θ_q matches Θ_h is,*

$$P(C_1 C_2 \cdots C_n | B_1 B_2 \cdots B_n) = \frac{(1-\alpha) k_d}{n_v} \left[(1-\alpha) \frac{k_d}{n_b} \right]^{n-1} \tag{3.14}$$

Proof. Apply Bayesian equation, we have

$$P(C_1C_2\cdots C_n|B_1B_2\cdots B_n) = \frac{P(B_1B_2\cdots B_n|C_1C_2\cdots C_n)P(C_1C_2\cdots C_n)}{P(B_1B_2\cdots B_n)}. \tag{3.15}$$

Note that $P(B_1B_2\cdots B_n|C_1C_2\cdots C_n)$ is the conditional probability that a correct matched sequence survive n hypothesis tests for each corresponding pair $(\mu_{\theta_{q,k}}, \mu_{\theta_k})$. Since, these tests are independent due to independent sensor noises, we have

$$P(B_1B_2\cdots B_n|C_1C_2\cdots C_n) = (1-\alpha)^n. \tag{3.16}$$

Joint probability $P(C_1C_2\cdots C_n)$ refers to the unconditional probability of being correct locations. We know that $P(C_1) = 1/n_v$ given there are n_v possible solutions. Then

$$P(C_1C_2) = P(C_2|C_1)P(C_1) = \frac{1}{n_b}\frac{1}{n_v}$$

because C_2 can only be chosen from neighbors of C_1. Extending this process by induction, we have

$$P(C_1C_2\cdots C_n) = \frac{1}{n_b^{n-1}}\frac{1}{n_v} \tag{3.17}$$

Since each node has equal and independent probability of being one of k_d readings, we have $P(B_k) = 1/k_d$ for $k = 1, \cdots, n$, we have,

$$P(B_1B_2\cdots B_n) = \frac{1}{k_d^n}. \tag{3.18}$$

Plugging (4.20), (4.21), and (4.22) into (5.9), we obtain the lemma. □

Remark 1. *Lemma 1 reveals important results about when the localization scheme works*

and how efficient it would be. If $P(C_1C_2\cdots C_n|B_1B_2\cdots B_n)$ increases as n increases, then the algorithm would eventually find the correct location. The rate of increase determines localization speed. This is largely determined by $(1-\alpha)\frac{k_d}{n_b}$.

Quantity $(1-\alpha)\frac{k_d}{n_b}$ is determined by sensor accuracy, the map property, and the trajectory. Let us take a close look. n_b is the expected number of neighbors and mostly remains as a constant since most intersections are 4-way intersections. If a map contains many different road headings, then

$$(1-\alpha)\frac{k_d}{n_b} > 1$$

and $P(C_1C_2\cdots C_n|B_1B_2\cdots B_n)$ increases as n increases. In such cases, the localization will be swift. However, if the map is purely rectilinear, then $k_d = n_b$ is the worst case scenario which leads to a decreasing $P(C_1C_2\cdots C_n|B_1B_2\cdots B_n)$. The algorithm fails in such cases. Fortunately, most maps are not completely rectilinear [62].

Since the hypothesis test setup is very conservative in rejection, we might end up with many candidate solutions in the process. To address the problem and find the most possible candidates, we classify the computed probabilities of (5.9) into two groups using the Ostu method [63] which selects the global optimal threshold by maximizing the between-class variance. If the group with higher probability only has one candidate then the vehicle is localized. Otherwise, it means that the group with higher probability contains several trajectories with higher probabilities which indicate more observations are needed to localize the vehicle. The number of solutions is the group size.

3.3.4 Map Entropy Analysis

Since quantity $(1-\alpha)\frac{k_d}{n_b} > 1$ is a necessary condition for localization feasibility and it also determines the localization efficiency, it is important to know how well it stands in real world. Note that ultimately, this is related to the value of k_d, the number of possible headings. To measure the heading variation, we introduce the Shannon entropy to mea-

sure road heading distributions [62]. We will experimentally investigate how information entropy in heading impacts localization performance in Section 4.4.

We divide orientation from 0° to 360° into $n_\mathtt{j}$ bins, and represent the orientation range set by $\{O_\mathtt{j} | \mathtt{j} = 1, 2, \cdots, n_\mathtt{j}\}$. In our set up, $n_\mathtt{j} = 36$. Let $\rho_\mathtt{j}$ be the relative frequency that θ_i in $O_\mathtt{j}$

$$\rho_\mathtt{j} = \sum_{v_i \in \mathcal{V}_{h,l}} I(\theta_i \in O_\mathtt{j})/n_v, \tag{3.19}$$

where $I(A) = 1$ if A is true and $I(A) = 0$ if false. The entropy of road orientation of $\mathcal{V}_{h,l}$ is

$$H(\mathcal{V}_{h,l}) = -\sum_j \rho_\mathtt{j} \log_{n_j} \rho_\mathtt{j}. \tag{3.20}$$

Fig. 3.4 shows the entropy of different maps. As for road maps with grid-like streets, e.g. Manhattan, NY (Fig. 3.4 (c)), the constructed HG has lower entropy than those with street orientation in all directions, e.g. College Station, TX (Fig. 3.4 (a)). Based on data from [64], we obtained entropies for 100 cities around the world analyzed in Fig. 3.4(d). 75 cities have entropies higher than 0.90 and only 5 cities have entropies lower than 0.70. This entropy histogram helps us predict how likely it is that our algorithm will be successful, which will be shown in Section 4.4.

3.4 Experiments

We have implemented our algorithm using MATLAB under a PC with Windows 7. Here we first investigate how HG entropy affects PLAM performance using simulation in 4.4.1.2. We then evaluate real world maps on simulated vehicle trajectories and lastly test our proposed method in physical experiments.

3.4.1 Simulated Maps with Different Entropies

Here we simulate maps with different entropies and study the relationship among HG entropy, number of query observations (i.e. n in (5.2)), and number of solutions.

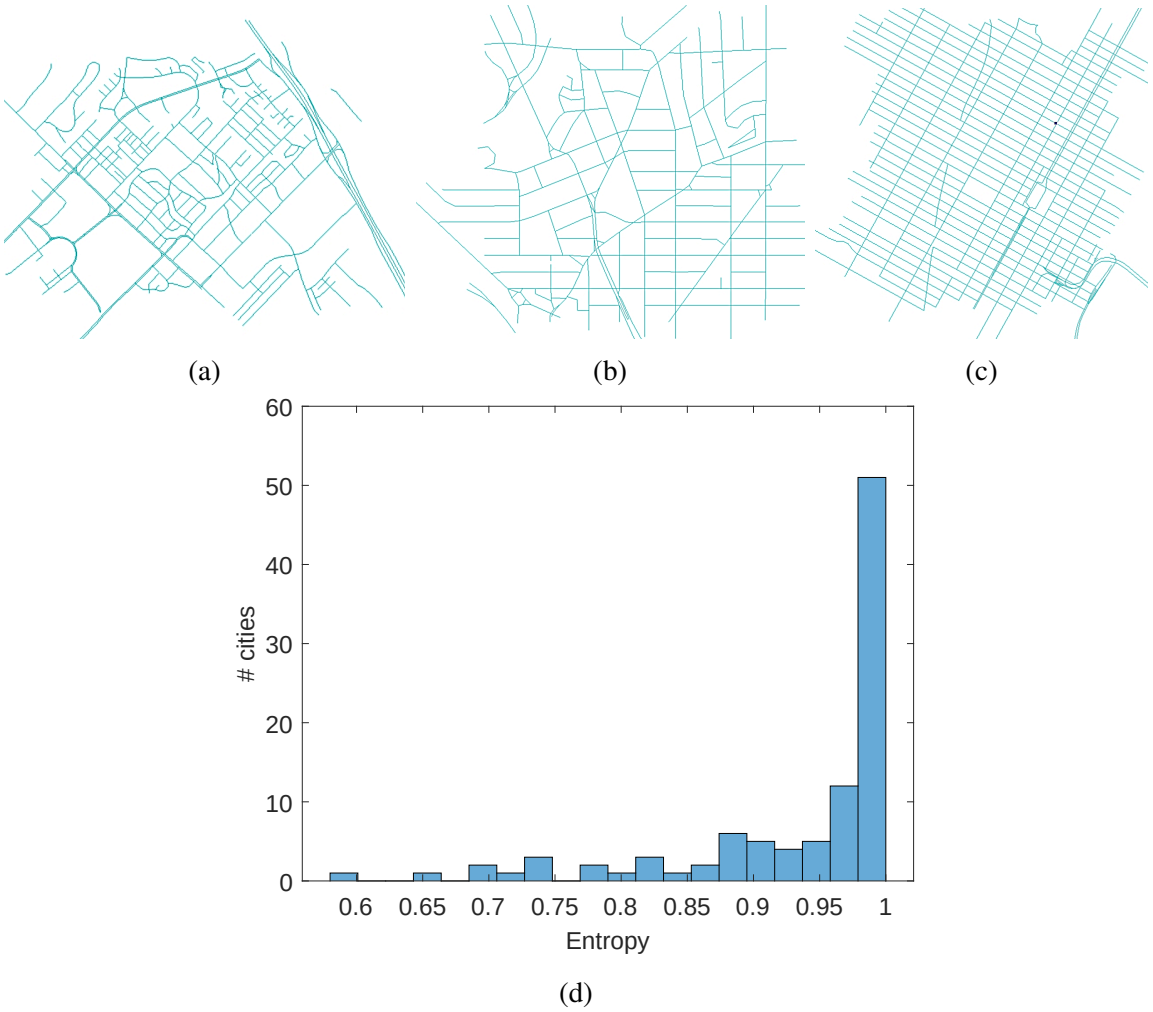


Figure 3.4: HG entropies for (a) College Station, TX, (b) Washington D.C., (c) Manhattan, NY, and (d) entropy distribution for 100 cities. Reprinted with permission from [1] © 2019 IEEE.

Based on the entropy range in Fig. 3.4(d), we generate 40 maps with entropy ranging from 0.60 to 0.99. We start with generating a square grid street map with 15×15 road intersections and 420 straight bi-directional roads. The map setup leads to a heading graph with 840 vertices and 2334 edges. With the fixed HG structure, we increase the entropy level by randomly selecting roads and perturbing their orientations. Figs. 3.5(a) and 3.5(b) show examples of a low entropy map and a high entropy map, respectively. On each of

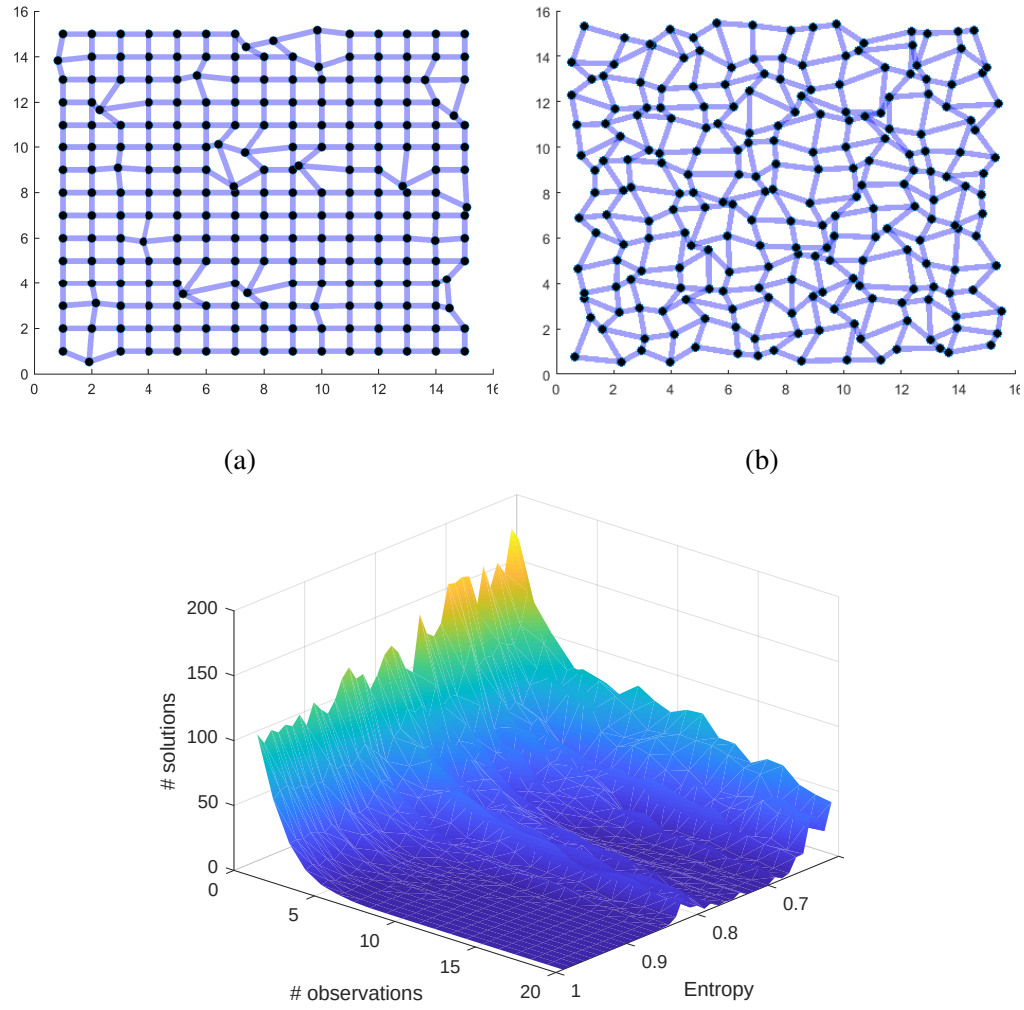


Figure 3.5: (a) Simulated map with low entropy (entropy=0.60). (b) Simulated map with high entropy (entropy=0.99). (c) Average #solutions with respect to map entropies and #observations. Reprinted with permission from [1] © 2019 IEEE.

the 40 simulated maps, we generate 20 query sequence samples with $n = 1, \cdots, 20$ with orientation standard deviation $\sigma_{\theta_{q,k}} = 5°$. To show the simulation result of #solutions with respect to map entropies and #observations, we take the average of results from 20 query sequence samples to obtain results shown in Fig. 3.5(c).

As expected, when the HG has a low entropy in orientation, it is likely to have multi-

ple solutions which fail to uniquely localize the vehicle even given more observations. Of course, these candidate solutions can be useful if combined with other localization methods. When the map entropy is above 0.9, #solutions converges to 1 quickly as n increases. Fig. 3.5(c) shows if the map entropy is above 0.9, the vehicle can be localized with $n \leq 10$. This covers a majority of cities according to Fig. 3.4(d). By analyzing the entropy of HG, we can evaluate if the proposed algorithm is applicable for localization task in advance.

3.4.2 Physical Experiments

Table 3.1: MAP INFO. AND #OBSERVATIONS n FOR LOCALIZATION. © 2019 IEEE.

Maps	Size (km^2)	Drivable road (km)	Entropy	# nodes	n
KITTI00	4.75	44.2	0.972	583	10,5
KITTI05	3.24	43.7	0.950	548	4,5
CSMap1	3.24	52.7	0.994	483	6,6
CSMap2	5.75	92.43	0.994	1102	4,6

We evaluate real world maps on simulated vehicle trajectories and test our proposed method using real world data from physical experiments. The real world maps are obtained from OSM. We tested maps from two cities:

- CSMap: College Station, Texas, U.S.. We have two test maps: CSMap1 and CSMap2.
- KITTI: Karlsruhe, Germany. We have two test maps: KITTI00 and KITTI05.

The first 4 columns of Tab. 4.1 describe the map size, the total length of drivable roads, entropy, and # nodes in the constructed HG for each map. On each map, similar to the process in Sec. 4.4.1.2, we generate 100 query sequence samples with $n = 1, \cdots, 20$ and orientation noise $\sigma_{\theta_{q,k}} = 5^\circ$ and the averaged results regarding both mean #solutions and

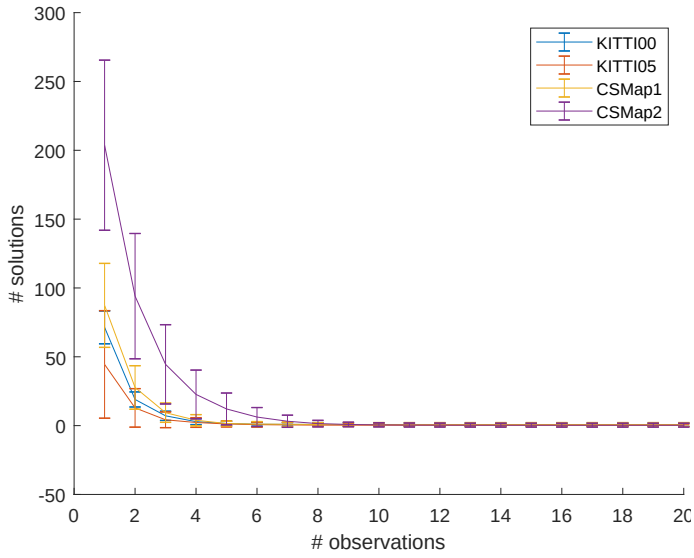


Figure 3.6: The localization speed for four maps, which are measured by number of observation needed to find a unique solution. Reprinted with permission from [1] © 2019 IEEE.

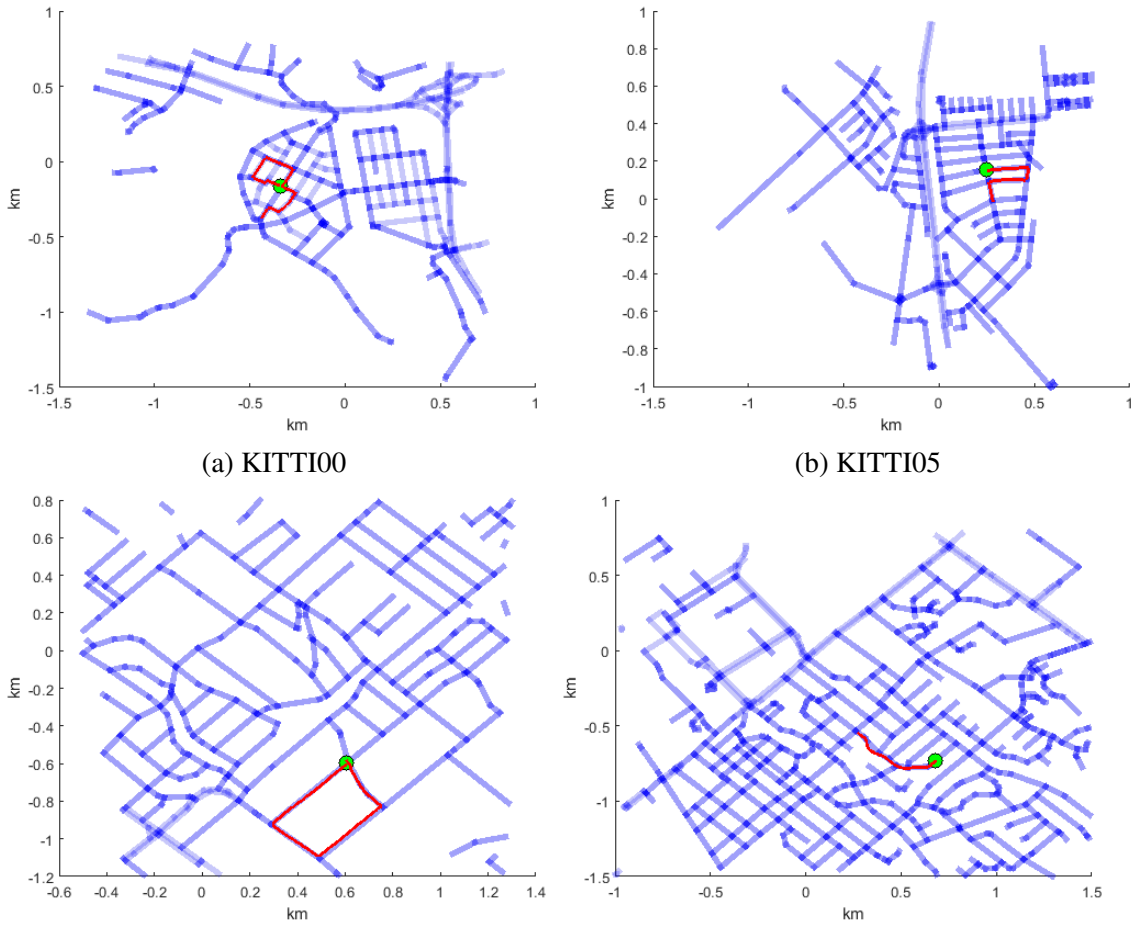
their corresponding standard deviations are visualized in Fig. 3.6. The entropy range of the four maps is 0.95 to 0.995. The results in Fig. 3.6 show the vehicle can be localized using no more than 10 observations which is consistent with the results in simulated maps (Fig. 3.5(c)).

We then test on query sequence collected from cars. We have sequences from both CSMap and KITTI. We obtain heading information from both IMU and compass readings. The two datasets are:

- CSData: We collect the query data using a Google PIXEL phone which consist of gyroscope readings at 400Hz and compass readings at 50Hz.
- KITTI: The KITTI dataset [65] consists of images recorded by an autonomous driving platform with ground truth of camera trajectory provided by a high-grade GPS-INS system. We use the gyroscope readings (100Hz) in INS as relative orientation input. To get the absolute heading, we only use the GPS readings to synthesize

compass readings to test our algorithm.

We summarize the results in the last column of Tab. 4.1. We have two sample trajectories For each map and a total of eight trials. Four typical sample trajectories are shown in Fig. 3.7. In all tests, the localization succeeds using no more than 10 observations.



(a) KITTI00

(b) KITTI05

(c) CSMap1

(d) CSMap2

Figure 3.7: Sample results from physical experiments. Candidate positions are green nodes. The map region is shown in blue color and the vehicle trajectory is shown in red line. (Best viewed in color) Reprinted with permission from [1] © 2019 IEEE.

3.5 Conclusion

We reported our PLAM method that did not rely on the perception and recognition of landmarks to localize robots/vehicles in urban environments. The proposed method intended to serve as a fall back solution when everything else fails due to bad weather or other environmental challenges. The method only employed a gyroscope and a compass along with a prior geographical map which was a low cost solution. PLAM pre-processed geographic maps into a heading graph which stores all long and straight segments of road as nodes. The information from the sensors was used to obtain heading changes during traveling. To localize the robot, the heading sequence was matched to the heading graph in a Bayesian framework that tracks both sensor and map uncertainties. Since the degenerated maps with a rectangular grid-like pattern may cause localization failure, we introduced information entropy to investigate map properties and studied how the PLAM method perform under maps with different entropies. Our results found that the PLAM method can serve as an effective localization method for a majority of cities.

4. GRAPH-BASED PROPRIOCEPTIVE LOCALIZATION USING A DISCRETE HEADING-LENGTH FEATURE SEQUENCE MATCHING APPROACH*

While the PLAM method in Chapter 3 is able to localize the vehicle using only heading sequence, the localization is only intermittent for turns. Also, it suffers if the vehicle travels in a degenerated map (e.g. rectilinear environments). This motivates us to design a new method enables continuous localization by considering inertial measurement units (IMUs) and wheel encoder inputs and is less limited by map degeneracy (e.g. rectilinear environments). Inspired by biological systems, we combine proprioceptive sensors, such as IMUs and wheel encoders, with magnetoreception, to develop a map-based localization method to address the problem, which is named as graph-based proprioceptive localization (GBPL).

In a nutshell, our new GBPL method employs the proprioceptive sensors to estimate vehicle trajectory and match it with a prior known map. However, this is non-trivial because 1) there is a significant drift issue in the dead reckoning process and 2) the true vehicle trajectory does not necessarily match the street GPS waypoints on the map due to the fact that a street may contain multiple lanes and street GPS waypoints may be inaccurate. This determines that a simple trajectory matching would not work. Instead, we focus on matching features which are straight segments of the trajectory (Fig. 4.1). We keep track of connectivity, heading and length of each segment which converts the trajectory to a discrete and connected query sequence. This allows us to formulate the GBPL problem as a probabilistic graph matching problem. To facilitate the Bayesian graph matching, we pre-process the prior known map consisting of GPS waypoints into a heading-length graph

*Reprinted with permission from "Graph-based Proprioceptive Localization Using a Discrete Heading-Length Feature Sequence Matching Approach" by Hsin-Min Cheng and Dezhen Song. *IEEE Transactions on Robotics (T-RO)*, Copyright © 2020 IEEE.

(HLG) to capture the connectivity of straight segments and their corresponding heading and length information. As the robot travels, we perform sequential Bayesian probability estimation until it converges to a unique solution. With global location obtained, we track robot locations continuously and align the trajectory with HLG to bound error drift.

We have implemented our algorithm and tested it in physical experiments using our own collected data and an open dataset. The algorithm successfully and continuously localizes the robot. The experimental results show that our method outperforms in localization speed and robustness when compared with the counterpart in [1]. The algorithm achieves localization accurate at the level that the prior map allows (less than 10m).

4.1 Related Work

Our GBPL is related to localization using different sensor modalities, dead-reckoning, and map-based localization.

We can classify the localization methods into two categories based on sensor modalities: exteroceptive sensors and proprioceptive sensors. Exteroceptive sensors mainly rely on the perception and recognition of landmarks in the environment to estimate location. Mainstream exteroceptive sensors include cameras [14, 19, 20] and lidars [24–26]. These methods are often challenged by poor lighting conditions or weather conditions. GPS receiver [27, 29] is another commonly-used sensor but it malfunctions when the vehicle travels close to high-rise buildings or inside tunnels. On the other hand, proprioceptive sensors, such as IMUs [4] and wheel encoders [31], are inherently immune to external conditions. However, they are more susceptible to error drift and suffer from limited accuracy. Recent sensor fusion approaches that combine an exteroceptive sensor, such as a camera or a laser ranger finder, with a proprioceptive sensor such as an IMU, greatly improve system robustness and become popular in applications [5]. However, the sensor fusion approaches still strongly depend on exteroceptive sensor and cannot handle the

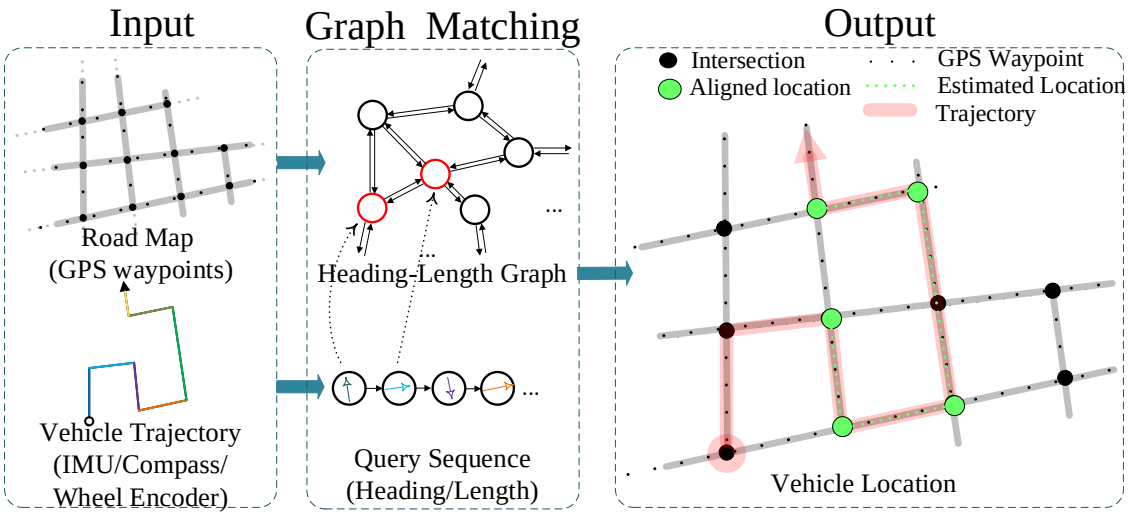


Figure 4.1: An illustration of GBPL method. Left: our inputs include a prior known map and the trajectory estimated from an IMU, a compass, and a wheel encoder. Middle: we process the prior map in to a straight segment connectivity graph and also the trajecory into a query sequence of headings and lengths of straight segments. Right: Aligned trajectory to the map after graph matching. Reprinted with permission from [2] Copyright © 2020 IEEE.

aforementioned challenging conditions.

To utilize proprioceptive sensors for navigation, dead reckoning integrates sensor measurements to compute robot/vehicle trajectory. The sensor measurements often include readings from accelerometers, gyroscopes, and/or wheel encoders [33]. There are many applications using the dead reckoning approach such as autonomous underwater vehicles (AUVs) [34] and pedestrian step measurement [30, 35]. To estimate the state of the robot/vehicle, filtering-based schemes such as unscented Kalman filter (UKF) [36] and particle filter (PF) [37, 38] are frequently employed. However, the nature of dead reckoning causes it to inevitably accumulate errors over time and lead to significant drift. To reduce the error drift, different methods have been proposed such as applying velocity constraint on wheeled robots [39] and modeling the wheel slip for skid-steered mobile robots [33]. These approaches have reduced error drift but cannot remove it completely.

Error still accumulates over time and causes localization failure. To fix the issue, we will show that drift can be bounded to map accuracy level by using map matching if the filtering-based approach with graph matching are combined.

Our method is a map-based localization [19,40–43]. According to [12], map representation can be classified into two categories: the location-based and the feature-based. The location-based maps are represented with specific locations of objects. For example, those existing geographic maps consisted of coordinate of locations such as OpenStreetMaps™ (OSM) [44] and Google Maps [45]. Geographic maps have been widely used to improve upon GPS measurements and there are common measures being used such as point-to-point, point-to-curve, curve-to-curve matching or advanced techniques [46]. The feature-based map is consisted with features of interest with its location. An example is ORB features [48] for visual simultaneous localization and mapping. In this work, we extract heading-length graph from geographic maps which converts a location-based map to a feature-based map to facilitate robust localization which also reduces graph size to speed up computation in the process.

Closely-related works include [40, 66, 67], which focus on map-aided localization using proprioceptive sensors for mobile robots. In [66], only vehicle speed and speed limit information from map are used as a minimal sensor setup. However, known initial position is required and the method achieves an accuracy of around 100 meters. In [40], the velocity from wheel encoder and steering angles are used for odometry and a particle filter based map matching scheme helps estimate vehicle positions. It does not consider velocity errors from the wheel encoder such as slippery or inflation levels. In [67], odometer and gyroscope readings are used for extended Kalman filter (EKF)-based dead reckoning and a map is used to correct errors when driving a long distance or turning at road intersections. The average positional error is 5.2 meters, but it again requires an initial position from GPS. It is worth noting that our localization solution does not require a known initial

position.

This paper is a significant improvement over our early work [1] where only heading sequence is used and localization is only intermittent for turns. The new method enables continuous localization by considering wheel encoder inputs and is less limited by map degeneracy (e.g. rectilinear environments). Also, we bound error drift in location alignment and verification after graph matching.

4.2 Problem Formulation

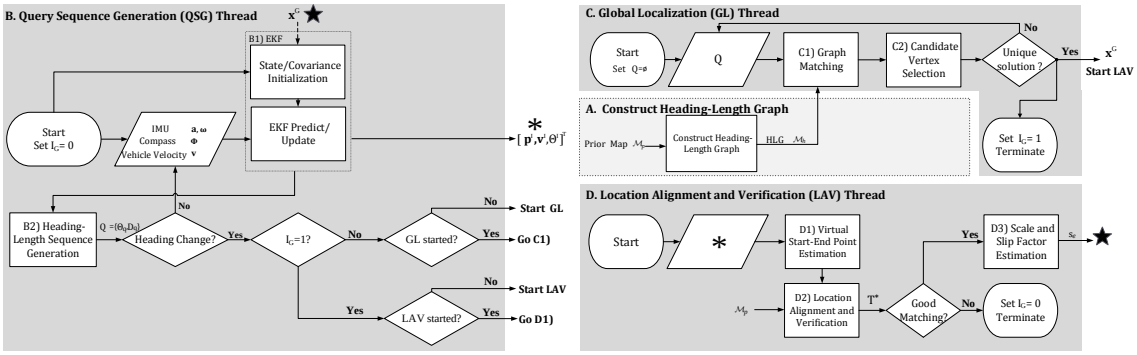


Figure 4.2: GBPL System Diagram. Reprinted with permission from [2] © 2020 IEEE.

4.2.1 Scenarios and Assumptions

In our set up, a robot or a vehicle (We interchangeably use "robot" and "vehicle.") is navigating in a poor weather conditions such as a severe thunderstorm or a whiteout snowstorm. No other exteroceptive sensors work properly. However, it is still necessary for the vehicle to find its location.

The vehicle/robot is equipped with an IMU, a digital compass or a magnetometer, and an on-board diagnostics (OBD) scanner which provides velocity feedback while navigating in an area with a given prior road map, e.g. OpenStreetMaps (OSM) [44]. We have the

following assumptions:

a.0 The vehicle is able to navigate in the environment and make turns at appropriate locations. If needed, the vehicle is willing to change its course by making additional turns to assist our algorithm to find its location.

a.1 The prior road map contains straight segments in most part of its streets and streets are not strict grids with equal side lengths.

a.2 The robot is a nonholonomic system,. i.e. it only performs longitudinal motion without lateral or vertical motions.

a.3 The IMU and the compass are co-located, pre-calibrated, and fixed at the vehicle geometric center.

a.4 The IMU, compass, and velocity readings are synchronized and time-stamped.

As part of the input of the problem, a prior road map consisting of a set of roads with GPS waypoints is required. The typical distance between adjacent waypoints is around 20m.

4.2.2 Nomenclature

Common notations are defined as follows,

- $\mathcal{M}_p := \{\mathbf{x}_m = [x_m, y_m]^\mathsf{T} \in \mathbb{R}^2 | m \in \mathscr{M}\}$ represents the prior road map which is a set of GPS positions where $\mathscr{M}$ is the position index set. Note that these GPS positions are map points instead of live GPS inputs. We do NOT use GPS receiver in our algorithm design.

- $\mathbf{a} = \{\mathbf{a}_j \in \mathbb{R}^3 | j = 0, 1, \cdots, N_j\}$ and $\omega = \{\omega_j \in \mathbb{R}^3 | j = 0, 1, \cdots, N_j\}$ denote accelerometer readings and gyroscope angular velocities from the IMU, respectively.

- $\phi = \{\phi_{j_\phi} \in \mathbb{R} | j_\phi = 0, \cdots, \lfloor \frac{N_j}{c_\phi} \rfloor\}$ denotes compass readings where $c_\phi \geq 1$ since a compass often has lower sampling frequency than that of the IMU.

- $\mathbf{v} = \{v_{j_v} \in \mathbb{R} | j_v = 0, \cdots, \lfloor \frac{N_j}{c_v} \rfloor\}$ denotes wheel speed readings from OBD where $c_v \geq 1$ because it has a lower sampling frequency than that of IMU. And v_{j_v} is the speed at midpoint of car rear wheels.

- $\mathcal{M}_h = \{\mathcal{V}_h, \mathcal{E}_h\}$ denotes the HLG where $\mathcal{V}_h$ is the vertex set and $\mathcal{E}_h$ and is the edge set.

- $Q = \{\Theta_q, D_q\}$ denotes the query heading-length sequence which consists of the segmented heading-length sequence. Θ_q is the set of heading sequence and D_q is the set of travel length sequence.

- $\mathcal{C}_k$ represents the candidate vertex set where $k = 1, \cdots, n$ is the length of the query sequence.

The GBPL problem is defined as follows.

Problem 2. *Given $\mathcal{M}_p$, $\mathbf{a}$, ω, ϕ and $\mathbf{v}$, localize the robot after its heading changes. As its localized, report robot location continuously.*

4.3 GBPL Modeling and Design

Our system diagram is illustrated in Fig. 5.2 which consists of four main building blocks: HLG construction, query sequence generation (QSG) thread, global localization (GL) thread, and location alignment and verification (LAV) thread. HLG construction is shaded in light gray which converts the prior geographic map into an HLG which runs only once in advance. For the rest shaded in dark gray, we refer to them as threads because they can be implemented as a parallel multi-threaded application. The QSG thread runs EKF constantly at the back end as the system receives sensory readings $\mathbf{a}$, ω, ϕ and $\mathbf{v}$

and outputs the estimated trajectory. GL thread searches for the global location on a turn-by-turn basis. GL thread performs Bayesian graph matching between the query sequence extracted from the trajectory and the HLG. After the global location is obtained, GL terminates and LAV aligns the latest segment with the map and uses the result to rectify error drifting in the EKF in QSG. If no satisfying alignment is found, LAV terminates and the system restarts GL. In fact, GL thread and LAV thread work alternatively depending on whether the robot is localized or not. We begin with HLG construction.

4.3.1 HLG Construction

We pre-process map $\mathcal{M}_p$ to construct an HLG to facilitate heading-length matching. There are three reasons for using HLG instead of matching on $\mathcal{M}_p$ directly.

- First, the vehicle trajectory may not exactly match with $\mathcal{M}_p$. Since $\mathcal{M}_p$ and most maps do not have lane-level information, the discrepancy between the estimated trajectory and $\mathcal{M}_p$ is non-negligible which makes the direct trajectory-to-map matching unreliable. Fig. 4.3 shows an example. For the same route, the trajectories may be different due to driving on different lanes, driver habit, traffic, etc.

- Second, matching trajectory with $\mathcal{M}_p$ directly is computationally expensive because the searching space grows with the total number of GPS waypoint positions in $\mathcal{M}_p$.

- Third, the inevitably accumulated trajectory drift deteriorates the matching quality and makes the matching unreliable.

Therefore, we extract features from the map which are the long straight segments and represent them as the HLG. This leads to a graph matching approach that can mitigate the influence of the aforementioned three issues. We start with HLG construction based on our prior work [1] where we have estimated road curvature changes to capture orientation change and construct a heading graph (HG). Build on [1], we augment length information

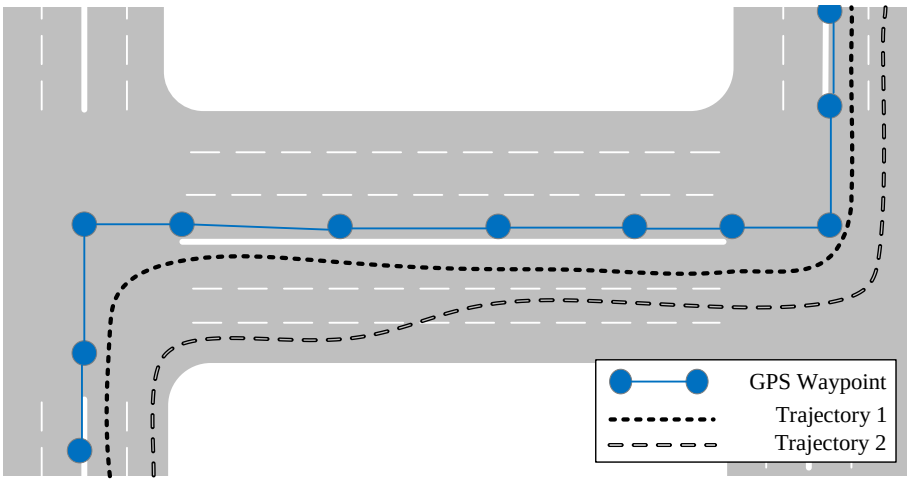


Figure 4.3: Map and trajectory discrepancy illustration. Given the trajectory generated by proprioceptive sensors, directly matching trajectory with the map may not be desirable. For the same route, trajectories 1 and 2 appear quite differently. Neither of them matches blue waypoints in the map. Reprinted with permission from [2] © 2020 IEEE.

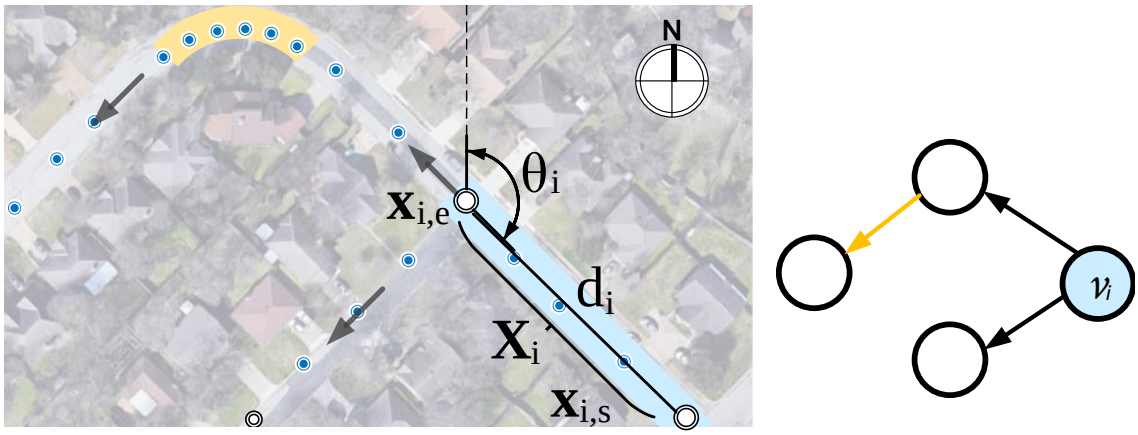


Figure 4.4: HLG illustration in color. The left figure shows a satellite image with road map consisted of GPS waypoints (blue dots) overlaying on top of the image and intersections represented in small black circles. We estimate road curvature changes to capture heading change and construct HLG. As an example, we color a long and straight segment with light blue and a curve segment with light orange. The right figure shows the corresponding HLG, and we only employ long road segment vertices for localization. Reprinted with permission from [2] © 2020 IEEE.

in HG to construct HLG for heading-length matching. Fig. 4.4 illustrates an example. For completeness, we provide an overview here and more detail description of constructing the graph can be found in [1]. The HLG $\mathcal{M}_h = \{\mathcal{V}_h, \mathcal{E}_h\}$ is a directed graph. A vertex $v_i \in \mathcal{V}_h$ represents a straight and continuous road segment with neither orientation changes nor intersections. An edge $e_{i,i'} \in \mathcal{E}_h$ captures the connectivity between nodes and characterizes the orientation change between the two connected vertices v_i and $v_{i'}$. $\mathcal{M}_h$ has two types of edges: road intersections and curve segments; and two types of vertices: long straight segment vertices and short transitional segment vertices. The short transitional segment vertices are often formed between curve segments or curved roads entering intersections.

To build $\mathcal{M}_h$, we split each road at road intersections and further segment them into two types of segments to capture orientation changes: straight segments and curved segments [1]. With all roads segmented, we compute orientation and length for vertices corresponding to those long straight road segments. Each vertex contains the following information

$$v_i = \{\mathbf{X}_i, \theta_i, d_i, b_i\}, \tag{4.1}$$

where $\mathbf{X}_i = [\mathbf{x}_{i,s}^\mathsf{T}, \cdots, \mathbf{x}_{i,e}^\mathsf{T}]^\mathsf{T}$ contained all 2D waypoint positions in GPS coordinates of the road segment with starting position $\mathbf{x}_{i,s}$ and ending position $\mathbf{x}_{i,e}$, orientation $\theta_i \in (-\pi, \pi]$ is the angle between the geographic north and the orientation of the road segment computed using $\mathbf{X}_i$ with a least squares estimation method adopted from [1], d_i is road segment length which is computed. by

$$d_i = ||\mathbf{x}_{i,s} - \mathbf{x}_{i,e}||, \tag{4.2}$$

and b_i is the binary variable indicate if the vertex is a long road segment. We only perform

orientation estimation if $d_i > t_l$ where t_l is the threshold for road segment length. That is,

$$b_i = \begin{cases} 1, & d_i > t_l, \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

Only long road segments ($b_i = 1$) will be used in localization which defines vertex subset $\mathcal{V}_{h,l} \subseteq \mathcal{V}_h$ corresponding to long straight segments. Note that θ_i depends on the robot traveling direction and hence $\mathcal{M}_h$ is a directed graph.

The errors of GPS waypoints in each entry of X_i affect the accuracy of θ_i and d_i. To track map uncertainties caused by GPS errors, we derive the distribution of θ_i and d_i using error variance propagation analysis [56]. We model GPS errors by using Gaussian distribution and assuming GPS measurement noises to be independent and identically distributed. We denote the GPS measurement variance by σ_g^2. According to [19], typical consumer grade navigation systems offer positional accuracy around $\sigma_g = 10$m. The distribution of θ_i that characterizes its uncertainty is

$$\theta_i \sim \mathcal{N}(\mu_{\theta_i}, \sigma_{\theta_i}^2), \tag{4.4}$$

where $\sigma_{\theta_i}^2$ is derived in [1]. And the distribution of d_i is

$$d_i \sim \mathcal{N}(\mu_{d_i}, \sigma_{d_i}^2) = \mathcal{N}(\mu_{d_i}, 2\sigma_g^2). \tag{4.5}$$

4.3.2 Query Sequence Generation (QSG) Thread

To localize the vehicle on $\mathcal{M}_h$, we estimate the trajectory from sensory readings with an EKF-based approach. We then generate a discrete query consisting of a heading-length sequence extracted from the EKF trajectory results. It is worth noting that our method is

not sensitive to the global drift of the EKF estimated trajectory because we only use short segmented trajectory to extract heading and length of its straight segments.

4.3.2.1 EKF-based Trajectory Estimation

Note that readings from the IMU, the digital compass, and the vehicle velocity: $\mathbf{a}$, $\boldsymbol{\omega}$, ϕ, and $\mathbf{v}$, are the inputs to the EKF-based approach to estimate vehicle trajectory [57–59]. To start the EKF, we need a stabilized initial compass reading ϕ_0 to determine the initial vehicle orientation which can be obtained by driving on a long and straight segment of road (Assumption a.2). We define two right-handed coordinate systems: IMU/compass device body frame $\{B\}$ (also overlapping with vehicle geometric center), the fixed inertial frame $\{I\}$ which shares its origin with $\{B\}$ at the initial pose. Frame $\{I\}$'s X-Y plane is a horizontal plane parallel to the ground plane with Y axis pointing to magnetic north direction and Z axis is vertical and points upward. In the state representation, let state vector $\mathbf{X}_{s,j}$ at time j be:

$$\mathbf{X}_{s,j} := [\mathbf{p}_j^I, \mathbf{v}_j^I, \boldsymbol{\Theta}_j^I, s_j]^\mathsf{T}, \tag{4.6}$$

which includes position $\mathbf{p}^I = [x, y, z]^\mathsf{T} \in \mathbb{R}^3$, velocity $\mathbf{v}^I = [\dot{x}, \dot{y}, \dot{z}]^\mathsf{T} \in \mathbb{R}^3$, and the Euler angles $\boldsymbol{\Theta}^I := [\alpha, \beta, \gamma]^\mathsf{T}$ in $\{I\}$ in X-Y-Z order, and scale/slip factor (SSF) s. We define s here to address vehicle velocity error which can be caused by tire radius error such as inflation level, road slippery, etc. The superscripts indicate in which frame the vector is defined. The transformation from $\{I\}$ to $\{B\}$ is the Z-Y-X ordered Euler angle rotation. The state transition equations are described as follows:

$$\begin{aligned}
\mathbf{p}_j^I &= \mathbf{p}_{j-1}^I + \tau_\omega \mathbf{v}^I \\
\mathbf{v}_j^I &= \mathbf{v}_{j-1}^I + \tau_\omega ({}_B^I\mathbf{R}(\mathbf{a}) - \mathbf{G}) \\
\Theta_j &= \Theta_{j-1} + \tau_\omega \, {}_B^I\mathbf{E}(\boldsymbol{\omega}) + \mathbf{c}_\gamma \\
s_j &= s_{j-1},
\end{aligned} \tag{4.7}$$

where τ_ω is the IMU sampling interval, $\mathbf{G} = [0\ 0\ -9.8]^\mathsf{T}$ is the gravitational vector, $\mathbf{c}_\gamma = [0\ 0\ \phi_0]^\mathsf{T}$ is the initial orientation determined by ϕ_0, ${}^I_B\mathbf{R}$ is the rotation matrix from $\{B\}$ to $\{I\}$, and ${}^I_B\mathbf{E}$ is the rotation rate matrix from $\{B\}$ to $\{I\}$.

For EKF observation models, we use velocity constraint from vehicle movement, sensory readings $\boldsymbol{\phi}$ and $\mathbf{v}$, and estimated scale by matching trajectory with map which will be discussed in Section 4.3.4.3. First, according to Assumptions a.2 there is no lateral or vertical movements in $\{B\}$, the velocities along Y axis and Z axis in $\{B\}$ are set to be zeros. The velocity constraint is written as:

$$({}^B_I\mathbf{R})_{2:3}\mathbf{v}^I_j = \begin{bmatrix} 0 & 0 \end{bmatrix}^\mathsf{T}, \tag{4.8}$$

where ${}^B_I\mathbf{R}_{2:3}$ is the second and third rows of ${}^B_I\mathbf{R}$.

From the coordinate definition, the heading direction is γ defined in $\{I\}$ (last component of $\boldsymbol{\Theta}^I$), we take compass reading $\boldsymbol{\phi}$ as its observation. In our physical system, compass readings have a lower sampling frequency than that of the IMU readings, we use the latest available reading. Also, compass readings may be polluted by other magnetic fields, we can recognize faulty readings by cross-validating compass readings with IMU readings. We discard the faulty compass readings if the difference between the estimated heading state and the compass reading exceeds an threshold. With the cross-validated compass reading, we update heading direction γ by

$$\gamma_j = \begin{cases} \phi_{j_\phi}, & \text{if } j = c_\phi j_\phi \\ \phi_{j_\phi - 1}, & \text{otherwise.} \end{cases} \tag{4.9}$$

We compensate SSF s_j by estimating its value from aligned map data after taking a turn. We will detail how to compute s_{ssf} and its variance in Section 4.3.4.3. For s, we

have

$$s_j = s_{ssf}, \tag{4.10}$$

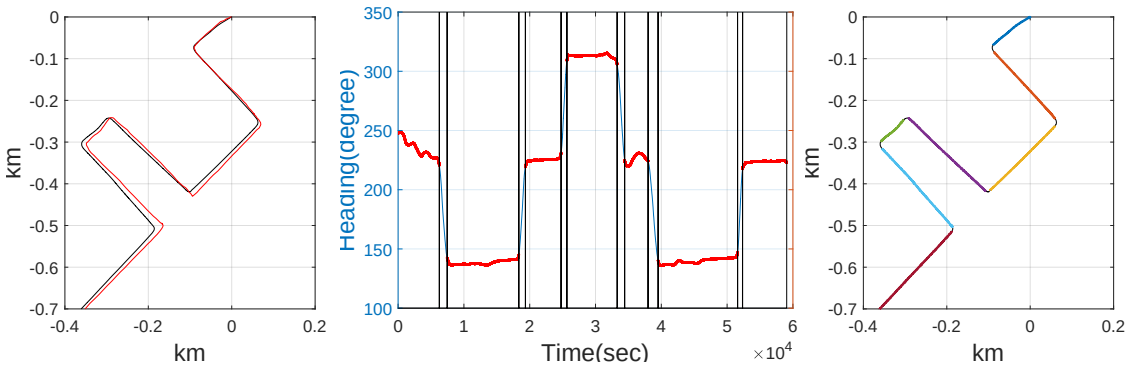
where s_{ssf} is the ratio of the trajectory length from the map versus that from the query. Lastly, we take wheel velocity $\mathbf{v}$ as observations. Similar to ϕ that the sampling frequency is lower than IMU readings, we have

$$||\mathbf{v}_j^I|| = \begin{cases} s_j v_{j_v}, & \text{if } j = c_v j_v \\ s_j v_{j_v - 1}, & \text{otherwise.} \end{cases} \tag{4.11}$$

Combining (4.9), (4.8), (4.11), and (4.10), we complete the observation model functions. The rest is to follow the standard EKF setup. Fig. 4.5(a) shows the estimated EKF trajectory compared with the corresponding GPS ground truth trajectory. Note that the vehicle takes some additional turns to assist localization (Assumption a.0) and the trajectory is not the shortest.

4.3.2.2 Heading-Length Sequence Generation

With the estimated trajectory, we generate query heading-length sequence by capturing vehicle heading changes. We adopt the method for heading sequence generation from [1] and augment corresponding length sequence in this work. To improve the robustness, we only keep headings when the vehicle is traveling on long and straight road segments. This means the headings should be stable and constant in a long stretch of travel time and corresponding travel distance is long. From the coordinate definition, the headings is γ in $\{I\}$ and is denoted by $\gamma_{0:j}$. To obtain the query sequence, we segment $\gamma_{0:j}$ to get stable headings and remove false positive headings that do not correspond to long and straight road segments. In Fig. 4.5(b), red horizontal segments are detected stable headings. Hence we obtain the set of query heading sequence which is denoted by $\Theta_q =$



(a) (b) (c)

Figure 4.5: (a) Trajectory estimation result: the red line is the GPS ground truth, and black line illustrates the EKF estimated trajectory. (b) Query heading representations. Blue line is estimated heading, black vertical lines are indices where data segmented, red lines mark out stable heading segments and unmarked segments are detected turns. (c) Corresponding travel heading and length segment representations. Different segments are marked in different colors. Reprinted with permission from [2] © 2020 IEEE.

$\{\Theta_{q,k}|k = 1, \cdots, n\}$ where k is query data index, n is the number of straight segments. Each subset $\Theta_{q,k}$ corresponding to continuous observations from EKF represents a straight segment. At the same time, we generate the corresponding travel length sequence which is denoted by $D_q = \{d_{q,k}|k = 1, \cdots, n\}$ where $d_{q,k}$ is the travel length of the segmented route (e.g. colored segments in Fig. 4.5(c)).

The query heading-length sequence $Q = \{\Theta_q, D_q\}$ is consists of the segmented heading-length sequence. The uncertainty of query sequence Q is obtained from EKF variance estimation. For Θ_q, we define $\overline{\theta_{q,k}}$ as the sample mean orientation of segment $\Theta_{q,k}$ which contains $n_{\theta_{q,k}}$ observations of random variable $\theta_{q,k}$. $\theta_{q,k}$ has it covariance matrix obtained from EKF. For D_q, the variance of $d_{q,k}$ can also be derived from EKF and we denote it by $\sigma^2_{d_{q,k}}$. Those variables will be used later in the analysis part.

It is worth noting that each entry of the sequence is not sensitive to the overall trajectory drift due to local trajectory segment computation. When segmenting into short segments,

the drift in each segment is smaller compare to the overall trajectory drift. The resulting sequence also can be understood as local features for the trajectory. Also, reducing the query to the discrete feature sequence helps in reducing computation complexity.

4.3.3 Global Localization Thread

4.3.3.1 GL Overview

With the query sequence obtained from on-board sensors, we are ready to match it with sequences on the HLG to search for the actual location. This is a graph matching problem. In the GL thread, we localize the robot when the robot changes its heading which is the moment the query sequence grows its length. It is worth noting that GL is an intermittent localization. The continuous localization will be address later in the paper.

Given the query heading-length sequence, we search for the best match of heading-length sequence in the HLG $\mathcal{M}_h$. For any long straight candidate vertex in $\mathcal{V}_{h,l}$, we match the query heading-length sequence with sequences of the vertices starting at the candidate vertex. We discard candidate vertices with poor matching. In each candidate sequence to query sequence matching, We model sensory and map uncertainties and formulate the matching process as a sequential hypothesis test problem. The result of GL depends on if a satisfying matching sequence can be found.

4.3.3.2 Graph Matching

The center part of GL is the matching of query sequence and candidate sequence on the graph. To achieve this, we expand the heading sequence matching in [1] to find the best heading-length matching in $\mathcal{M}_h$. Given query sequence $Q = \{\Theta_q, D_q\} = \{(\theta_{q,k}, d_{q,k})|k = 1, \cdots, n\}$, let us denote a candidate heading-length vertex sequence in $\mathcal{M}_h$ by $M := \{\Theta, D\} = \{(\theta_k, d_k)|k = 1, \cdots, n\}$ correspondingly. As a convention in this paper, for random vector $\star$, $\mu_\star$ represents its mean vector. Following the convention, mean matrix of Q is defined as $\mu_Q = [\mu_{\Theta_q}^\mathsf{T}, \mu_{D_q}^\mathsf{T}]^\mathsf{T}$ where $\mu_{\Theta_q} = [\mu_{\theta_{q,1}}, \cdots, \mu_{\theta_{q,n}}]^\mathsf{T}$ and

$\mu_{D_q} = [\mu_{d_{q,1}}, \cdots, \mu_{d_{q,n}}]^\mathsf{T}$. The mean matrix of M is denoted by $\mu_M = [\mu_\Theta^\mathsf{T}, \mu_D^\mathsf{T}]^\mathsf{T}$ where $\mu_\Theta = [\mu_{\theta_1}, \cdots, \mu_{\theta_n}]^\mathsf{T}$ and $\mu_D = [\mu_{d_1}, \cdots, \mu_{d_n}]^\mathsf{T}$.

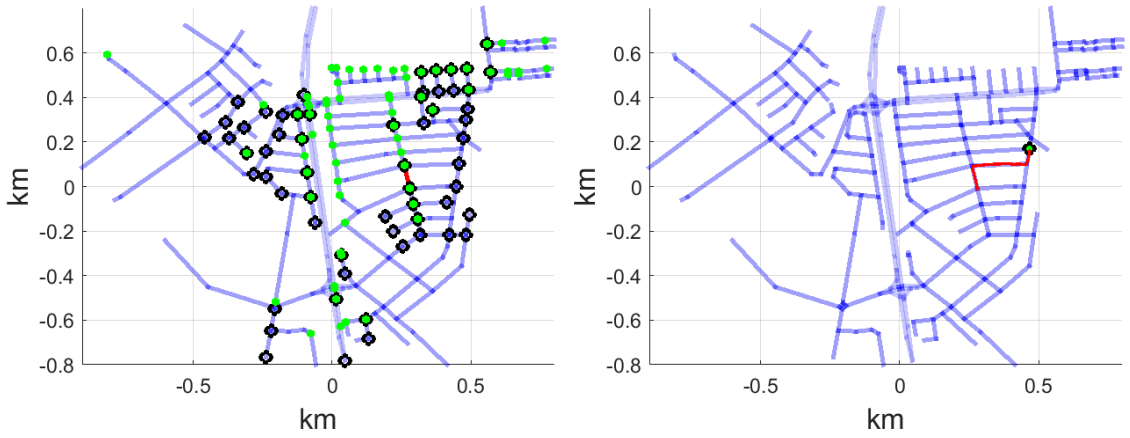
Due to independent measurement noises, the conditional matching probability between query sequence $Q := \{\Theta_q, D_q\}$ and a candidate sequence $M := \{\Theta, D\}$ on HLG $\mathcal{M}_h$ is

$$\begin{aligned} &P(\mu_Q = \mu_M | Q, M) \\ &= P(\mu_{\Theta_q} = \mu_\Theta | \Theta_q, \Theta) P(\mu_{D_q} = \mu_D | D_q, D). \end{aligned} \tag{4.12}$$

From [1], the conditional heading matching probability between Θ_q and Θ_h is

$$P(\mu_{\Theta_q} = \mu_\Theta | \Theta_q, \Theta) \propto \prod_{k=1}^{n} f_T(t(\theta_{q,k}, \theta_k)), \tag{4.13}$$

due to independent sensor noises and $f_T(t(\theta_{q,k}, \theta_k))$ is the probability density function



(a) (b)

Figure 4.6: An example of global localization. (a) The candidate locations using heading matching (green dots), length matching (black circle). We show that performing heading-length matching (locations with green dot and black circle) helps reducing candidates. (b) The candidate localization is reduced to the single solution if the joint distribution between heading and length is used. Reprinted with permission from [2] © 2020 IEEE.

(PDF) of Student's t-distribution. For length matching, the conditional matching probability between D_q and D is

$$P(\mu_{D_q} = \mu_D | D_q, D) \propto \prod_{k=1}^{n} f(z(d_{q,k}, d_k)), \tag{4.14}$$

where $f(\cdot)$ is the PDF of standard normal distribution, and $z(d_{q,k}, d_k) = \frac{d_{q,k} - d_k}{\sqrt{\sigma^2_{d_k} + \sigma^2_{d_{q,k}}}}$. Combining (4.13) and (4.14) and recalling that n is the number of straight segments in the query sequence, we rewrite (4.12) as follows,

$$P(\mu_Q = \mu_M | Q, M) \propto \prod_{k=1}^{n} f_T(t(\theta_{q,k}, \theta_k)) f(z(d_{q,k} - d_k)). \tag{4.15}$$

4.3.3.3 Candidate Vertex Selection

To select on candidate vertices during matching, we perform statistical hypothesis testing to remove unlikely matchings. According to (4.12), sequence matching is considered as multiple pair matching. For each pair $(\{\theta_k, d_k\}, \{\theta_{q,k}, d_{q,k}\})$, it is a hypothesis testing

$$\begin{aligned} \mathbf{H_0}: &\quad [\mu_{\theta_{q,k}}, \mu_{d_{q,k}}]^\mathsf{T} = [\mu_{\theta_k}, \mu_{d_k}]^\mathsf{T} \\ \mathbf{H_1}: &\quad \text{otherwise.} \end{aligned} \tag{4.16}$$

Hypothesis H_0 can be seen as two null hypotheses: $H_{0,\theta} : \mu_{\theta_{q,k}} = \mu_{\theta_k}$ and $H_{0,d} : \mu_{d_{q,k}} = \mu_{d_k}$. We perform two individual tests separately with significance level $1 - \alpha$ where α is a small probability. Both $H_{0,\theta}$ and $H_{0,d}$ are two-tailed distributions. We choose $t_{\alpha/2,\nu}$ as the t-statistic with a cumulative probability of $(1 - \frac{\alpha}{2})$ where ν is the degrees of freedom (DoF) and $z_{\alpha/2}$ as the z-statistic with a cumulative probability of $(1 - \frac{\alpha}{2})$. We reject H_0 if

$$(|t(\theta_k, \theta_{q,k})| > t_{\alpha/2,\nu}) \vee (|z(d_k, d_{q,k})| > z_{\alpha/2}). \tag{4.17}$$

By sequentially applying the hypothesis testing on each corresponding pair $(\{\theta_k, d_k\}, \{\theta_{q,k}, d_{q,k}\})$ from query sequence Q and candidate sequence M on HLG $\mathcal{M}_h$, we determine whether M represents the actual trajectory. Fig. 4.6 has shown that using the joint distribution of heading and length significantly reduce the number of solutions in the matching process.

In the matching process, we might get many candidate solutions because the hypothesis test is conservative in rejection. To address the problem and check if we converge to a unique solution, we classify the computed probabilities of (4.12) into two groups using the Ostu method [63]. The number of solutions is the group size. If the group with higher probability has only one candidate then the vehicle is localized. Otherwise, it means that the group with higher probability contains several trajectories with higher probabilities. It indicates that more observations are needed to localize the vehicle.

4.3.3.4 GL Algorithm

We summarize the heading-length matching method in Algorithm 2. In a nutshell, as we sequentially match the vertex down the query sequence, we compare it with the out-neighbor of remaining vertices on the graph using breadth-first search.

Note that vertex v_i may have adjacent vertices with same orientation. For example, consider the vehicle reaches a long straight road (with road intersections). This long straight road corresponds a set of vertices with same orientation. We denote the set of straight path start from v_i by $\mathcal{V}_s$.

To reuse the computed information as the query sequence grows, we define the candidate vertex information set $\mathcal{C}_k$ where $k = 1, \cdots, n$ is the length of the query sequence. The candidate vertex set is denoted by $\mathcal{C}_k = \{\{v_i, \mathcal{V}_{M,i}, p_i\} | i = 1, \cdots, n_{\mathcal{C}_k}\}$, where each element in $\mathcal{C}_k$ record the candidate vertex $v_{\mathtt{i}}$ (the starting vertex of the trajectory/path), $\mathcal{V}_{M,i}$ is the set of vertex path, and the matching probability p_i in (4.12) and $n_{\mathcal{C}_k}$ is the cardinality of $\mathcal{C}_k$. To initialize, we set $\mathcal{C}_0 := \{\{v_i, \emptyset, \frac{1}{|V_{h,l}|}\} | i = 1, \cdots, |\mathcal{V}_{h,l}|\}$ because each

vertex in $\mathcal{V}_{h,l}$ is equally likely to be the path starting vertex. The computational complexity of calculating each term in (4.12) is $O(1)$ using the alias sampling method [68]. The upper bound of candidate vertex cardinality is $|\mathcal{V}_{h,l}|$ and thus it takes $O(|\mathcal{V}_{h,l}|)$ to compute probability of all candidate vertices. The size of straight path set takes $O(|\mathcal{V}_s|)$ which is related to variation of map road headings in Sec. 4.3.3.6. With little variation in headings (e.g. Manhattan streets), $|\mathcal{V}_s|$ is larger. On the contrary, $|\mathcal{V}_s|$ is small compared to $|\mathcal{V}_{h,l}|$ with large variation in road headings. In this case, $O(|\mathcal{V}_s|) = O(1)$. The classification of probabilities into two groups is $O(|\mathcal{V}_{h,l}|)$ using Hoare's selection algorithm.

We summarize the computational complexity of Algorithm 2 in Lemma 6.

Lemma 2. *The computation complexity of the heading-length matching is* $O(n|\mathcal{V}_s||\mathcal{V}_{h,l}|)$.

4.3.3.5 Localization Analysis

The remaining problem is whether this sequence of hypothesis testing would converge to the true trajectory as the length of the sequence grows. To analyze this, let us define three binary events: $A_k = 1$ if $\mu_{d_{q,k}} = \mu_{d_k}$, $B_k = 1$ if $\mu_{\theta_{q,k}} = \mu_{\theta_k}$, and $C_k = 1$ if vertex k in M_h is the actual location. The joint event $C_1 \cdots C_n = 1$ is to say $M := \{\Theta, D\}$ represent the true trajectory, whereas we know $A_1 \cdots A_n B_1 \cdots B_n$ from sequence matching. In the analysis, we denote $n_v = |\mathcal{V}_{h,l}|$ as the cardinality of $\mathcal{V}_{h,l}$ and n_b as the expected number of neighbors for each vertex. We describe map/trajectory property in a rudimentary way by assuming k_d levels of distinguishable discrete headings in $[0, 2\pi)$ and k_l levels of distinguishable discrete road lengths. Each vertex takes a heading value and length value with equal probabilities of $1/k_d$ and $1/k_l$ correspondingly. Generally speaking, we know $n_v \gg k_d \geq n_b$ and $n_v \gg k_l \geq n_b$ for most maps. we have the following lemma.

Algorithm 1: Heading-length Graph Matching

Input: $\mathcal{M}_h = \{\mathcal{V}_h, \mathcal{E}_h\}$ and $Q = \{\Theta_q, D_q\}$

Output: $\mathcal{C}_k$ or vehicle location, I_G

1	$\mathcal{C}_0 := \{\{v_i, \emptyset, \frac{1}{\lvert V_{h,l}\rvert}\} \lvert i = 1, \cdots, \lvert\mathcal{V}_{h,l}\rvert\}$	$O(1)$
2	Initialize $I_G = 0$	$O(1)$
3	**for** $k = 1, \cdots, n$ **do**	$O(n)$
4	$\quad \mathcal{C}_k \leftarrow \emptyset$;	$O(1)$
5	$\quad$ **for** $i = 1, \cdots, n_{C_{k-1}}$ **do**	$O(\lvert\mathcal{V}_{h,l}\rvert)$
6	$\quad\quad$ **if** $k == 1$ **then**	
7	$\quad\quad\quad$ Access straight path set $\mathcal{V}_s$ start from v_i;	$O(1)$
8	$\quad\quad$ **else**	
9	$\quad\quad\quad v_{i'} \leftarrow$ last vertex in path $\mathcal{V}_{M,i}$	$O(1)$
10	$\quad\quad\quad V_{i'} \leftarrow$ adjacent verteices of $v_{i'}$ (with different angles);	$O(1)$
11	$\quad\quad\quad$ Access straight path set $\mathcal{V}_s$ start from each vertex in $V_{i'}$;	$O(1)$
12	$\quad\quad$ **for** $V_s \in \mathcal{V}_s$ **do**	$O(\lvert\mathcal{V}_s\rvert)$
13	$\quad\quad\quad$ Access θ_s and d_s of V_s;	$O(1)$
14	$\quad\quad\quad$ compute $p \leftarrow f_T(t(\theta_s, \theta_{q,k}))f(z(d_s, d_{q,k}))$	$O(1)$
15	$\quad\quad\quad$ **if** *Pass hypothesis testing in* (4.16) **then**	
16	$\quad\quad\quad\quad$ Update matching probability $p_{i'} \leftarrow p_i \cdot p$	$O(1)$
17	$\quad\quad\quad\quad V_{M,i'} \leftarrow$ Append $\mathcal{V}_s$ to $\mathcal{V}_{M,i}$	$O(1)$
18	$\quad\quad\quad\quad \mathcal{C}_k \leftarrow \mathcal{C}_k \cup \{v_i, \mathcal{V}_{M,i'}, p_{i'}\}$	$O(1)$
19	$\quad$ Classify probabilies in (4.12) of C_k using Otsu's method;	$O(\lvert\mathcal{V}_s\rvert\lvert\mathcal{V}_{h,l}\rvert)$
20	$\quad$ Remove group in C_k with lower probabilities;	$O(1)$
21	$\quad$ **if** $\lvert C_k\rvert > 1$ **then**	
22	$\quad\quad$ Return C_k;	$O(1)$
23	$\quad$ **else**	
24	$\quad\quad$ Set $I_G = 1$;	$O(1)$
25	$\quad\quad$ Return vehicle location;	$O(1)$

Lemma 3. *The conditional probability that $M = \{\Theta, D\}$ is the true matching sequence given that $Q = \{\Theta_q, D_q\}$ matches M is,*

$$P(C_1 \cdots C_n | A_1 \cdots A_n B_1 \cdots B_n) = \frac{(1-\alpha)^2 k_d k_l}{n_v} \left[(1-\alpha)^2 \frac{k_d k_l}{n_b}\right]^{n-1} \tag{4.18}$$

Proof. Applying the Bayesian equation, we have

$$P(C_1 \cdots C_n | A_1 \cdots A_n B_1 \cdots B_n) = \frac{P(A_1 \cdots A_n B_1 \cdots B_n | C_1 \cdots C_n) P(C_1 \cdots C_n)}{P(A_1 \cdots A_n B_1 \cdots B_n)}. \tag{4.19}$$

Indeed $P(A_1 \cdots A_n B_1 \cdots B_n | C_1 \cdots C_n)$ is the conditional probability that a correct matched

sequence survives n hypothesis tests in (4.16). Due to independent measurement noises, we have $P(A_1B_1|C_1) = (1-\alpha)^2$. Besides, these tests are independent due to independent sensor noises, we have

$$P(A_1 \cdots A_n B_1 \cdots B_n | C_1 \cdots C_n) = (1-\alpha)^{2n}. \tag{4.20}$$

Joint probability $P(C_1 \cdots C_n)$ is actually the unconditional probability of being correct locations. We know $P(C_1) = 1/n_v$ given there are n_v possible solutions, and $P(C_2|C_1) = 1/n_b$ because there are n_b neighbors of C_1. By induction,

$$P(C_1 \cdots C_n) = \frac{1}{n_b^{n-1}} \frac{1}{n_v}. \tag{4.21}$$

Lastly, each vertex takes a heading value and length value with equal and independent probabilities of $1/k_d$ and $1/k_l$. We have $P(A_k B_k) = \frac{1}{k_d k_l}$ and

$$P(A_1 \cdots A_n B_1 \cdots B_n) = \frac{1}{(k_d k_l)^n}. \tag{4.22}$$

Plugging (4.20), (4.21), and (4.22) into (5.9), we obtain the lemma. □

Corollary 1. *We have shown in [1] that the conditional probability that Θ is the true matching given Θ_q is*

$$P(C_1 \cdots C_n | B_1 \cdots B_n) = \frac{(1-\alpha)k_d}{n_v} \left[(1-\alpha) \frac{k_d}{n_b} \right]^{n-1} \tag{4.23}$$

Compare (4.18) *with* (4.23)*, we have*

$$\frac{P(C_1 \cdots C_n | A_1 \cdots A_n B_1 \cdots B_n)}{P(C_1 \cdots C_n | B_1 \cdots B_n)} = [(1-\alpha)k_l]^n \tag{4.24}$$

Since $k_l > \frac{1}{1-\alpha}$ is generally true, localization using both heading and length information $Q = \{\Theta_q, D_q\}$ is faster than using heading Θ_q only.

Under assumption a.1, Lemma 7 shows the probabilistic convergence of right matching. Also, it reveals when the localization scheme works and localization efficiency. If $P(C_1C_2\cdots C_n|A_1A_2\cdots A_nB_1B_2\cdots B_n)$ increases as n increases, the proposed method would find the correct location eventually. The localization speed is determined by $(1-\alpha)^2\frac{k_dk_l}{n_b}$ which is determined by sensor accuracy, the map property, and the trajectory [1]. Since n_b (the expected number of neighbors) remains constant as most intersections are 4-way intersections, k_d and k_l (spreading in heading and length) are the main factors determining the increasing rate of $P(C_1C_2\cdots C_n|A_1A_2\cdots A_nB_1B_2\cdots B_n)$. If a map contains many different road headings and lengths, then $P(C_1C_2\cdots C_n|A_1A_2\cdots A_nB_1B_2\cdots B_n)$ increases swiftly as n increases. On the contrary, if the map only contains purely rectilinear grids then $k_d = n_b$ and $k_l = 1$. This is the worst case scenario which leads to a decreasing $P(C_1C_2\cdots C_n|A_1A_2\cdots A_nB_1B_2\cdots B_n)$ and the algorithm fails. Fortunately, most maps do not have the issue [64]. If a rectilinear map has different side lengths in each distribute, the algorithm still works (assumption a.1). To better understand how it stands in real world, we analyze map proprieties in the following section.

4.3.3.6 Map Entropy Analysis

To provide a measure of variation and spreading in heading and road length, we introduce the Shannon information entropy to measure road heading and length distributions [69]. To minimize the effect of bin size on calculated entropy, we set orientation bin widths to be 5°, and 20 meters for road length. Let us denote orientation range set by $\{O_{\mathtt{j}}|\mathtt{j} = 1, 2, \cdots, n_{\mathtt{j}}\}$ and length range set by $\{L_{\mathtt{i}}|\mathtt{i} = 1, 2, \cdots, n_{\mathtt{i}}\}$. We define $n_{ji} = n_jn_i$ and $\rho_{\mathtt{ji}}$ be the relative frequency that $\theta_i \in O_{\mathtt{j}}$ and $d_i \in L_{\mathtt{i}}$. The joint Shannon

entropy in heading and road length is

$$H_{\theta,d}(\mathcal{V}_{h,l}) = -\sum_{\mathtt{j}}\sum_{\mathtt{i}} \rho_{\mathtt{ji}} \log_{n_{\mathtt{ji}}} \rho_{\mathtt{ji}}. \tag{4.25}$$

By analyzing the entropy of different maps, we predict localization efficiency of our algorithm, which will be shown in Section 4.4.

4.3.4 Location Alignment and Verification Thread

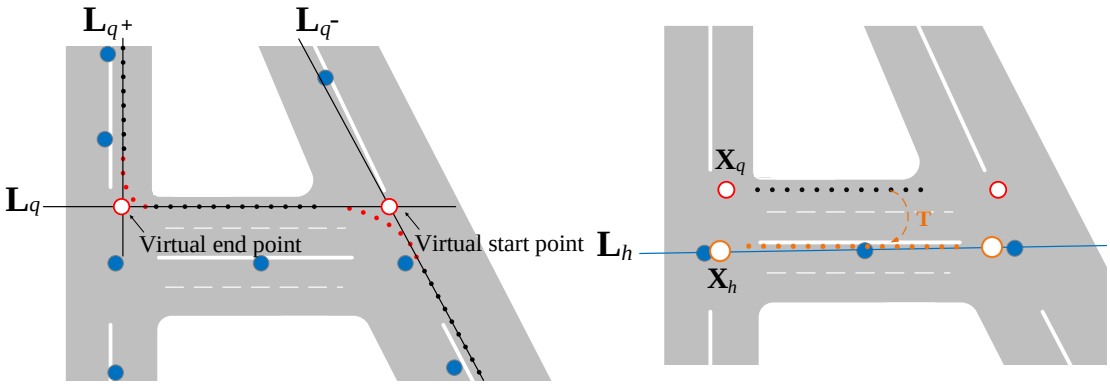


Figure 4.7: Illustration of LAV. The solid small dots represent vehicle trajectory where red points are turn points and black points belongs to SSPTE. The roads are shaded gray regions characterizing their width,and GPS waypoints in $\mathcal{M}_p$ are represented in larger blue dots. (a) Virtual start and end points (i.e. red circles) of an SSPTE. (b) Left: misalignment between $\mathbf{X}_q$ and $\mathbf{X}_h$. It is clear that SSPTM only has three points. Exact point-to-point matching is not appropriate. We fit a line $\mathbf{L}_h$ using SSPTM which is used as reference line for finding the best transformation between SSPTE and SSPTM points. Reprinted with permission from [2] © 2020 IEEE.

If the GL thread finds a unique position, we can start LAV thread to continuously report vehicle location. The key is to fix the EKF drift issue using the prior map information. This is achieved by monitoring if the vehicle makes a turn. Once a turn is identified, the

straight segment prior to the turn (SSPT) can be extracted. Comparing the SSPT from EKF estimation (SSPTE) to the corresponding SSPT on the map $\mathcal{M}_p$ (SSPTM), we can reset EKF parameters which rectifies the drifting issue.

Let us define the set of points in SSPTE by

$$\mathbf{X}_q = \{\mathbf{p}_\iota \in \mathbb{R}^2 | \iota = 1, \cdots, n_q\} \tag{4.26}$$

with each element obtained from EKF $\mathbf{p}^I_{1:2} = [x, y]^\mathsf{T}$ where $\mathbf{p}^I_{1:2}$ is the first and second element of $\mathbf{p}^I$. The distribution of $\mathbf{p}_\iota$ is $\mathbf{p}_\iota \sim \mathcal{N}(\mu_{\mathbf{p}_\iota}, \Sigma_{\mathbf{p}_\iota})$, where $\mu_{\mathbf{p}_\iota}$ is the mean vector and $\Sigma_{\mathbf{p}_\iota}$ is the covariance matrix obtained from the EKF. The corresponding GPS SSPTM points are defined by

$$\mathbf{X}_h = \{\mathbf{x}_l | l = 1, \cdots, n_h\} \tag{4.27}$$

and the covariance of GPS points is denoted by $\Sigma_g = diag(\sigma_g^2, \sigma_g^2)$ as mentioned in Section 4.3.1. We obtain $\mathbf{X}_h$ by using the localized position from GL thread and performing graph matching with the out-neighbor of vertices. Thus we have $\mathbf{x}_l \sim \mathcal{N}(\mu_{\mathbf{x}_l}, \Sigma_g)$.

4.3.4.1 Virtual Starting-Point and End-Point Estimation

However, SSPTE points do not necessary follow SSPTM as shown in Fig. 4.7(a). This is because we do not know which lane the vehicle is driving in and the map may not provide lane-level waypoint accuracy. Fig. 4.7(a) also shows the effect of vehicle turn radius which makes the length of SSPTE shorter than that of the corresponding SSPTM. To address the problem, we estimate virtual starting and end points for an SSPTE.

We find the virtual starting and end points by computing line intersection of two consecutive SSPTE segments. With the current segment positions $\mathbf{X}_q$, we denote the set of points from previous and next SSPTE segments by $\mathbf{X}_{q^-}$ and $\mathbf{X}_{q^+}$, respectively. Applying line fitting to $\mathbf{X}_q$, $\mathbf{X}_{q^-}$, and $\mathbf{X}_{q^+}$, we obtain three 2D lines $\mathbf{L}_q$, $\mathbf{L}_{q^-}$, and $\mathbf{L}_{q^+}$, respectively.

We parameterize each line by two reference points. Thus we denote $\mathbf{L}_q = [\mathbf{a}_q^\mathsf{T}, \mathbf{b}_q^\mathsf{T}]^\mathsf{T}$, $\mathbf{L}_{q^-} = [\mathbf{a}_{q^-}^\mathsf{T}, \mathbf{b}_{q^-}^\mathsf{T}]^\mathsf{T}$, and $\mathbf{L}_{q^+} = [\mathbf{a}_{q^+}^\mathsf{T}, \mathbf{b}_{q+}^\mathsf{T}]^\mathsf{T}$. Also, the line direction vectors are $\mathbf{v}_q = \mathbf{b}_q - \mathbf{a}_q$, $\mathbf{v}_{q^+} = \mathbf{b}_{q^+} - \mathbf{a}_{q^+}$, and $\mathbf{v}_{q^-} = \mathbf{b}_{q^-} - \mathbf{a}_{q^-}$. Finding the intersection between $\mathbf{L}_q$ and $\mathbf{L}_{q^-}$ allows us to obtain the virtual starting point. We denote the virtual starting point of $\mathbf{X}_q$ by $\mathbf{p}_s$.

$$\mathbf{p}_s = \mathbf{a}_q - \frac{\mathbf{v}_{q^-}^\perp.(\mathbf{a}_q - \mathbf{a}_q^-)}{\mathbf{v}_{q^-}^\perp.\mathbf{v}_q}\mathbf{v}_q, \tag{4.28}$$

where $\cdot$ is dot product and $\mathbf{v}_{q^-}^\perp$ is the perp operator of $\mathbf{v}_{q^-}$. Similarly, the intersection between $\mathbf{L}_q$ and $\mathbf{L}_{q^+}$ gives us the virtual end point $\mathbf{p}_e$. We have

$$\mathbf{p}_e = \mathbf{a}_q - \frac{\mathbf{v}_{q^+}^\perp.(\mathbf{a}_q - \mathbf{a}_q^+)}{\mathbf{v}_{q^+}^\perp.\mathbf{v}_q}\mathbf{v}_q, \tag{4.29}$$

where $\mathbf{v}_{q^+}^\perp$ is the perp operator of $\mathbf{v}_{q^+}$. When SSPTE is connected with an curve segment (e.g. caused by vehicle turn), we add $\mathbf{p}_s$ and $\mathbf{p}_e$ to $\mathbf{X}_q$ to help alignment process. $\mathbf{p}_s$ and $\mathbf{p}_e$ become the first and the last points in $\mathbf{X}_q$, respectively.

4.3.4.2 Location Alignment and Verification

With augmented $\mathbf{X}_q$, we can match $\mathbf{X}_q$ to $\mathbf{X}_h$ to rectify drifting issue by finding the transformation $\mathbf{T}$ between them (see Fig. 4.8). Here $\mathbf{T}$ is 3-DoF rigid body transformation represented by a 2x2 rotation matrix $\mathbf{R}$, and a 2x1 translation vector $\mathbf{t}$,

$$\mathbf{T}(\mathbf{x}) := \mathbf{R}\mathbf{x} + \mathbf{t}, \tag{4.30}$$

where $\mathbf{x}$ is a 2D point. $\mathbf{X}_q$ usually contains significantly more entries than that of $\mathbf{X}_h$ due to its higher sampling frequency ($n_q \gg n_h$). Directly matching two point sets is not the best solution. Instead, we fit a line through points in $\mathbf{X}_h$ and minimizing the distance of all points in $\mathbf{X}_q$ to this line (Fig. 4.7(b)).

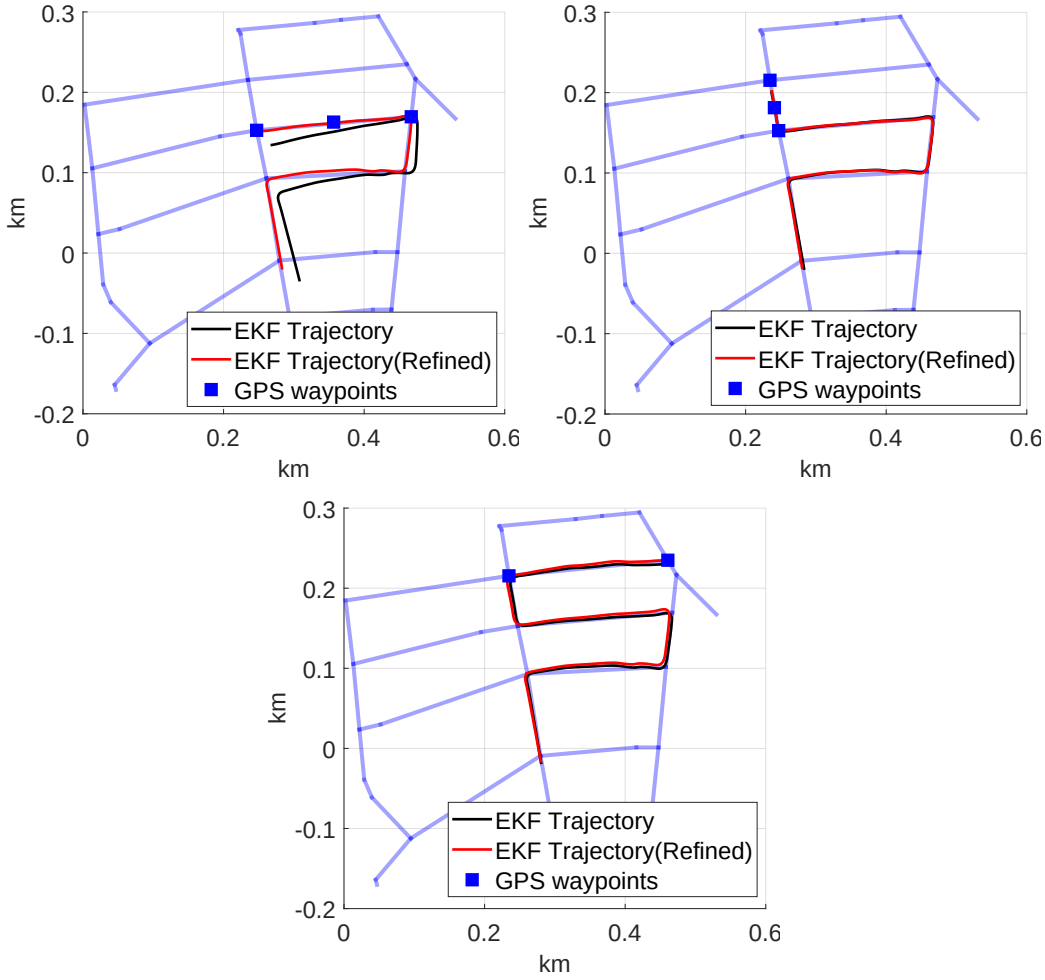


Figure 4.8: An example of location alignment and verification that keeps drifting under control where n is the number of long straight segments for the vehicle. The unaligned trajectory is shown in black, the aligned trajectory is shown in red, and GPS waypoints are shown in dark blue square. Adapted with permission from [2] © 2020 IEEE.

Let us denote $\mathbf{L}_h = [\mathbf{a}_h^\mathsf{T}, \mathbf{b}_h^\mathsf{T}]^\mathsf{T}$ where $\mathbf{a}_h$ and $\mathbf{b}_h$ are two reference points on the line. For every point $\mathbf{p}_j$ in $\mathbf{X}_q$, the point after transformation is denoted by $\mathbf{T}(\mathbf{p}_\iota)$. The point-to-line distance between $\mathbf{T}(\mathbf{p}_\iota)$ and $\mathbf{L}_h$ is defined as

$$d_\perp(\mathbf{T}(\mathbf{p}_\iota), \mathbf{L}_h) = \frac{||(\mathbf{a}_h - \mathbf{T}(\mathbf{p}_\iota) \times (\mathbf{a}_h - \mathbf{b}_h)||}{||\mathbf{a}_h - \mathbf{b}_h||}, \tag{4.31}$$

where '$\times$' is the cross product and $||\cdot||$ is the L^2 norm. We define the cost function $\mathbf{C_T}$ by

$$\mathbf{C_T} = \begin{bmatrix} d_\perp(\mathbf{T}(\mathbf{p}_s), \mathbf{L}_h) \\ d_\perp(\mathbf{T}(\mathbf{p}_1), \mathbf{L}_h) \\ \vdots \\ d_\perp(\mathbf{T}(\mathbf{p}_{n_q}), \mathbf{L}_h) \\ d_\perp(\mathbf{T}(\mathbf{p}_e), \mathbf{L}_h) \end{bmatrix}, \tag{4.32}$$

and formulate the following optimization problem

$$\arg\min_{\mathbf{T}} \mathbf{C}_\mathbf{T}^\mathsf{T} \Sigma_C^{-1} \mathbf{C_T} + \lambda ||\mathbf{T}(\mathbf{p}_s) - \mathbf{x}_1|| + \lambda ||\mathbf{T}(\mathbf{p}_e) - \mathbf{x}_{n_h}||, \tag{4.33}$$

where $\Sigma_C = diag(\sigma^2_{d_\perp,\mathbf{p}_s}, \cdots, \sigma^2_{d_\perp,\mathbf{p}_e})$, β is a nonnegative weight, and $\mathbf{x}_1$ and $\mathbf{x}_{n_h}$ are the first and the last entries in (4.27), respectively. $\sigma^2_{d_\perp,\mathbf{p}_\iota}$ is obtained using error propagation. In detail, let $d_\perp(\mathbf{T}(\mathbf{p}_\iota), \mathbf{L}_h) = f_d(\mathbf{p}_\iota, \mathbf{L}_h)$ and $\xi = [\mathbf{p}_s^\mathsf{T}, \mathbf{L}_h^\mathsf{T}]^\mathsf{T}$, we have $\sigma^2_{d_\perp,\mathbf{p}_\iota} = J_d \Sigma_d J_d^\mathsf{T}$, where $J_d = \frac{\partial f_d}{\partial \xi}$ and $\Sigma_d = diag(\Sigma_{\mathbf{p}_\iota}, \Sigma_{\mathbf{L}_h})$ because $\mathbf{p}_\iota$ is independent of $\mathbf{L}_h$ which comes from $\mathbf{X}_h$. Define $\mathbf{L}_h = f_L(\mathbf{X}_h)$, we have $\Sigma_{\mathbf{L}_h} = J_L \Sigma_{X_h} J_L^\mathsf{T}$ where $J_L = \frac{\partial f_L}{\partial \mathbf{X}_h}$ and $\Sigma_{X_h} = diag(\Sigma_g, \cdots, \Sigma_g)$. The second and third terms are soft constraints due to potential alignment errors. To solve (4.33), we start with a small positive weight for λ and apply a nonlinear optimization solver, e.g. Levenberg-Marquardt algorithm. Initially, we set $\mathbf{R} = \mathbf{I}_{2\times 2}$, and $\mathbf{t}$ from the result of the global location obtained from Section 4.3.3. For each turn, we use previous solution as the initial solution and increase λ gradually until the change in solution is negligible.

Now we have optimized $\mathbf{T}$ and we denote the aligned locations by $\hat{\mathbf{X}}_q = \mathbf{T}(\mathbf{X}_q)$. We need to verify if the matching result is reliable by performing hypothesis testing. We have

two hypotheses:

$$\begin{aligned} \mathbf{H_0} &: \mathbf{X}_h \text{ and } \hat{\mathbf{X}}_q \text{ are from the same distribution,} \\ \mathbf{H_1} &: \text{otherwise.} \end{aligned} \tag{4.34}$$

We set the significance level by α and reject H_0 if the statistic is less than α. Note H_0 is examined by the Mahalanobis distance $\mathbf{C}_\mathbf{T}^\mathsf{T}\Sigma_C^{-1}\mathbf{C}_\mathbf{T}$ which follows a χ^2 distribution with $2(n_q+2)$ DoFs. Thus we reject H_0 if $\mathbf{C}_\mathbf{T}^\mathsf{T}\Sigma_C^{-1}\mathbf{C}_\mathbf{T} > \chi^2_{2(n_q+2)}(\alpha)$. Correspondingly, we set localization status indicator variable I_G values by $I_G = \begin{cases} 0, & H_0 \text{ is rejected,} \\ 1, & \text{otherwise.} \end{cases}$ If $I_G = 1$, we accept $\mathbf{T}$ and use the aligned trajectory $\hat{\mathbf{X}}_q := \mathbf{T}(\mathbf{X}_q)$ which is used to reset the EKF states (Fig. 5.2). After LAV execution, we keep acquiring the vehicle locations EKF $\mathbf{p}^I_{1:2}$ until next turn. When turn is detected and $I_G = 1$, we execute LAV thread repeatedly. If $I_G = 0$, it means that we cannot find the position and we lose the global position. Thus we terminate the LAV thread and start the GL thread again. The possible reasons for losing global location could be the vehicle drives off the prior map or keep straight without turns which cause drifting too much.

4.3.4.3 SSF Estimation

To further reduce drift in the dead-reckoning process, we consider SSF in the EKF-based trajectory estimation. There are two sources of biases: systematic and non-systematic biases from wheel encoder inputs [70]. The systematic error can be caused by tire radius error such as inflation level, tire wear, gear ratio, etc. Non-systematic error comes from wheel slippage on road. To compensate for those errors, we introduce scale and slip factor s_{ssf} in (4.10).

To compute s_{ssf}, we need the travel length for each vertex on HLG for both query

data and map data. We obtain the travel length d_q on the query data using the virtual starting/end points $\mathbf{p}_e$ and $\mathbf{p}_s$ in (4.28) and (4.29). That is$d_q = ||\mathbf{p}_e - \mathbf{p}_s||$. According to (4.27), the corresponding travel length on the map is denoted by $d := ||\mathbf{x}_{n_h} - \mathbf{x}_1||$. Assuming GL thread ends at the n-th turn, for $k = (n+1), \cdots, n'$ we estimate s_{ssf} by computing the ratio of accumulated length $d_{q,k}$ and d_k:

$$s_{ssf} = \sum_{k=n+1}^{n'} d_k \Big/ \sum_{k=n+1}^{n'} d_{q,k}. \tag{4.35}$$

We then model the variance of s_{ssf} to be used in the EKF measurement variance in Section 4.3.2.1. It is not accurate to set a constant variance value for s_{ssf}, since at the beginning traveling length is short and thus s_e has larger variance. As the traveling length increases, the variance of s_{ssf} ought to decrease. Denote the variance of s_{ssf} by $\sigma^2_{s_{ssf}}$, we derive the following Lemma.

Lemma 4. *The variance of scale and slip factor s_{ssf} is*

$$\sigma^2_{s_{ssf}} = \frac{1}{L_q^2}(2n_s\sigma_g^2 + \frac{L_g^2}{L_q^2}\sum_{k=n+1}^{n'} \sigma^2_{d_q,k}). \tag{4.36}$$

Proof. First, we write s_{ssf} as function of measurements from d_k and $d_{q,k}$ according to (4.35). That is, $s_{ssf} = f_s(d_{n+1}, \cdots, d_{n'}, d_{q,n+1}, \cdots, d_{q,n'})$. We know the variance of d_k is $\sigma^2_{d_k} = 2\sigma_g^2$ from (4.5) and the variance of $d_{q,k}$ is $\sigma^2_{d_q,k}$ which is defined in Section 4.3.2.2. Let us define $L_q = \sum_{k=n+1}^{n'} d_{q,k}$, $L_g = \sum_{k=n+1}^{n'} d_k$, and $n_s = n' - n$. Through forward error propagation,

$$\sigma^2_{s_{ssf}} = J_s \Sigma_s J_s^{\mathrm{T}}, \tag{4.37}$$

where $\Sigma_s = diag(2\sigma_g^2, \cdots, 2\sigma_g^2, \sigma_{d_{q,n+1}}^2 \cdots \sigma_{d_{q,n'}}^2)$ and J_s is

$$\begin{aligned} J_s &= [\frac{\partial f_s}{\partial d_{n+1}}, \cdots, \frac{\partial f_s}{\partial d_{n'}}, \frac{\partial f_s}{\partial d_{q,n+1}}, \cdots, \frac{\partial f_s}{\partial d_{q,n'}}] \\ &= [\frac{1}{L_q} \cdots, \frac{1}{L_q}, \frac{-L_g}{L_q^2}, \cdots, \frac{-L_g}{L_q^2}]. \end{aligned} \tag{4.38}$$

Plug (4.38) into (4.37), we have

$$\begin{aligned} \sigma_{s_{ssf}}^2 = J_s \Sigma_s J_s^{\mathsf{T}} &= 2n_s \frac{\sigma_g^2}{L_q^2} + \sum_{k=n+1}^{n'} \sigma_{d_q,k}^2 \frac{L_g^2}{L_q^4} \\ &= \frac{1}{L_q^2}(2n_s\sigma_g^2 + \frac{L_g^2}{L_q^2} \sum_{k=n+1}^{n'} \sigma_{d_q,k}^2). \end{aligned} \tag{4.39}$$

□

Remark 2. *Let us take a close look at* (4.39)*. We have $L_q \approx L_g$ because the estimated travel length should be similar to the corresponding path in map. Therefore, we can approximate $\sigma_{s_{ssf}}^2$ as*

$$\sigma_{s_{ssf}}^2 = J_s \Sigma_s J_s^{\mathsf{T}} = \frac{1}{L_q^2}(2n_s\sigma_g^2 + \sum_{k=n+1}^{n'} \sigma_{d_q,k}^2).$$

Thus we show that $\sigma_{s_{ssf}}^2$ decrease as $L_q = \sum_{k=n+1}^{n'} d_{q,k}$ increases. As time goes, we have longer travel length and the estimation of s_{ssf} becomes more accurate. Using the accumulated travel length to adjust SSF is suitable to compensate systematic biases. If the traveling length is long and systematic biases are compensated, setting a sliding window for accumulated distance can be used to detect non-systematic biases that varies through traveling.

The resulting s_{ssf} and $\sigma_{s_{ssf}}^2$ are fed into the EKF in Section 4.3.2.1. This completes

our overall method.

4.4 Experiments

We have implemented the proposed GBPL method using MATLAB and validated the algorithm in both simulation and physical experiments. We first validate the proposed global localization approach. Second, we test the LAV performance.

For physical experiments, we evaluate our approach on three maps with seven outdoor data sets, as described below. We obtain the corresponding three maps from OSM:

- CSMap : College Station, Texas, U.S.
- KITTI00Map: Karlsruhe, Germany, and
- KITTI05Map: Karlsruhe, Germany.

Map information including map size, total length of drivable roads, HLG entropy, and #nodes in HLG is shown in the first four columns of Tab. 4.1.

The seven query sequences are three self-collected CSData sequences and four KITTI sequences:

- CSData: We record IMU readings at 400Hz and compass readings at 50Hz using a Google Pixel phone mounted on a passenger car. Also, we read the vehicle speed at 46.6Hz sampling frequency in average using a Panda OBD-II Dongle which provides the velocity feedback from vehicle wheel encoder. We have collected three sequences: CS-1, CS-2 and CS-3.
- KITTI: We use the KITTI GPS/IMU dataset [65] which contains synchronized IMU readings from its inertial navigation system (INS) as inputs. We only use the GPS readings to synthesize compass readings to test our algorithm since the data sets do not provide compass readings. We have four sequences: KITTI00-1, KITTI00-2, KITTI05-1, and KITTI05-2.

4.4.1 Global Localization Test

4.4.1.1 Evaluation Metrics and Methods Tested

It is worth noting that the speed of methods are characterized by n, number of straight segments in the query. Since computation speed is not a concern, we are more interested in how many inputs it takes to localize the vehicle. Therefore, n is a good metric for this. For a given n, the algorithms may provide multiple solutions if there is many similar routes in the map. If the number of solutions is one, then the vehicle is uniquely localized. The number of solutions is also an important measure for algorithm efficiency. Two algorithms are compared in our experiments:

- GBPL: Current method that uses both heading and length information of straight segments.
- PLAM: The counterpart method using heading only [1].

4.4.1.2 Map Entropy Evaluation

Map entropy describes how much the heading and distance distribution spread out in a given map. Higher entropy means distributions are more spread out and hence it is easier for the vehicle to localize itself, as proved in Lem. 7. Therefore, we want to find out what are map entropy range of real cities and use the range to test our GBPL. As shown in Fig. 4.9(a), we calculate map entropy distributions of 100 cites based on the data from [64].

Table 4.1: MAP INFO. AND #STRAIGHT SEGMENTS n FOR LOCALIZATION.

Maps	Size (km^2)	Drivable road (km)	Entropy	#nodes	n(PLAM)	n (GBPL)
CSMap	3.24	52.7	0.724	483	9,5,6	**3,3,2**
KITTI00Map	4.75	44.2	0.877	583	10,5	**4,3**
KITTI05Map	3.24	43.7	0.797	548	4,5	**3,4**

For comparison, the normalized sum of heading entropy and length entropy are in orange bars, and the heading entropy are in blue bars. For each city, the sum of heading entropy and length entropy is the upper bound of the joint entropy. We generate histogram plots for entropy distribution in Fig. 4.9(b) and Fig. 4.9(c). As shown in Fig. 4.9(c), 95 cities have entropy values higher than 0.70 and the lowest entropy is around 0.6. This determines that entropy range of maps that we will use to test our algorithm is from 0.60 to 0.99.

To better understand the relationship among HLG entropy, n, and the number of solutions, we simulate 40 maps with joint entropy of heading and length ranging from 0.60 to 0.99. Building on the simulation in [1], we expand it from Heading Graph to HLG in this work. For completeness, we repeat information about experimental settings here. The simulated maps are with a fixed graph structure, and we increase the entropy level in both heading and length by perturbing selected road intersection positions. For each map, we generate 20 query sequence samples with $n = 1, \cdots, 20$ and the uncertainties of orientation and length are considered by setting $\sigma_{\theta_{q,k}} = 5^\circ$, $\sigma_{d_{q,k}} = \sqrt{2}\sigma_g$, and $\sigma_g = 5$ meters. We compute the number of solutions by averaging the results of 20 sequences for each map. The simulation result is shown in Fig. 4.10(b) and we adapt Fig. 4.10(a) from [1] for comparison.

For PLAM which uses heading only (Fig. 4.10(a)), the vehicle can be localized with $n \leq 10$ if the entropy in orientation is above 0.9 [1]. Under GBPL, the vehicle can be localized with $n \leq 7$ even if the heading/length entropy is 0.6. It is worth noting that lower entropy means less spreading of heading and segment length and road network is closer to be a rectilinear grid and hence it is more challenging to localize a vehicle in such settings. GBPL appears to be more robust to low map entropy than PLAM.

Fig. 4.10(a) and Fig. 4.10(b) show the number of solutions with regard to n values and different HLG entropy values. We fix the entropy as 0.87 and $n = 3$ in Figs. 4.10(c) and 4.10(d), respectively to observe how quickly the number of solutions decreases in each

setting. It shows the #solutions decreases more rapidly in GBPL than that of PLAM using heading only. This result is consistent with Cor. 1.

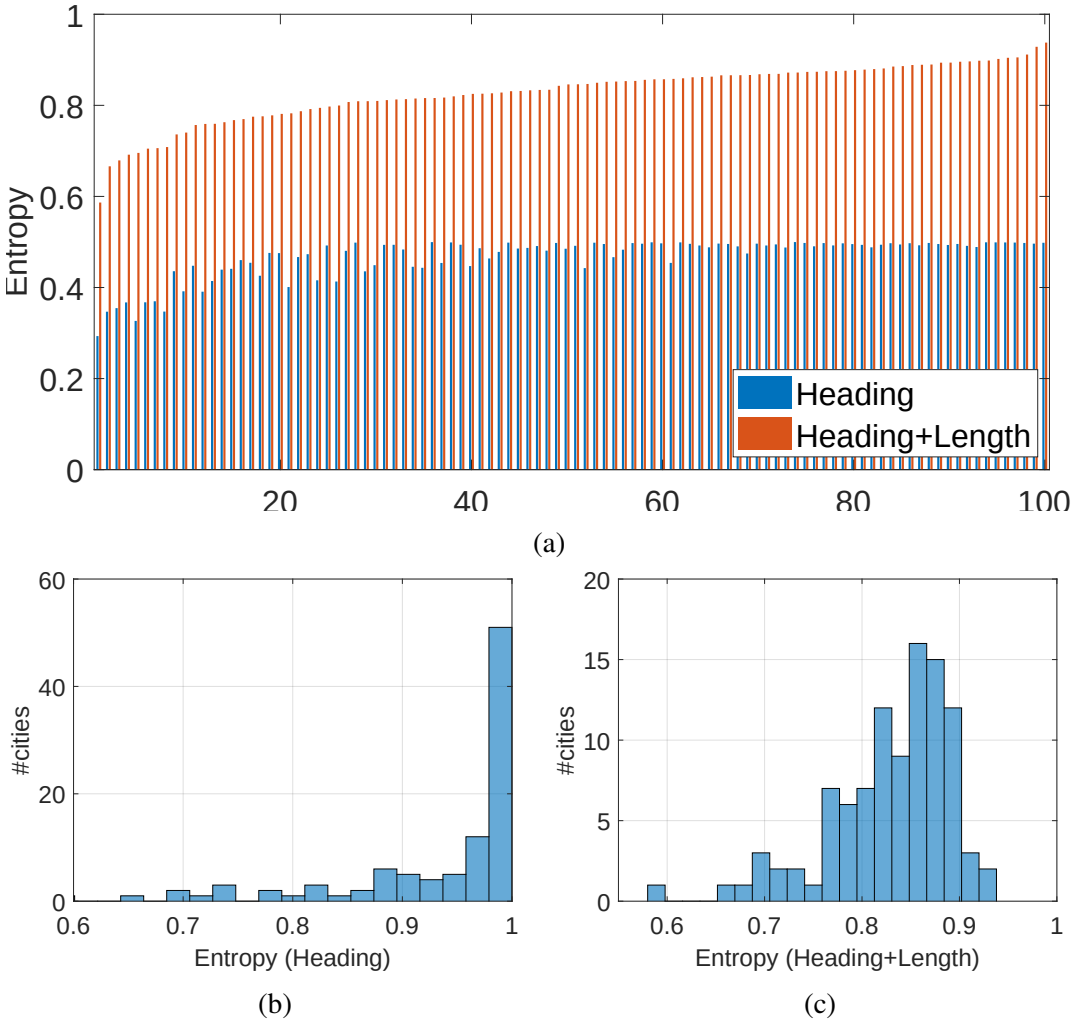


Figure 4.9: (a) Entropy of 100 cities. (b) Heading entropy distribution of 100 cities. (c) Heading and length entropy distribution of 100 cites. Adapted with permission from [2] © 2020 IEEE.

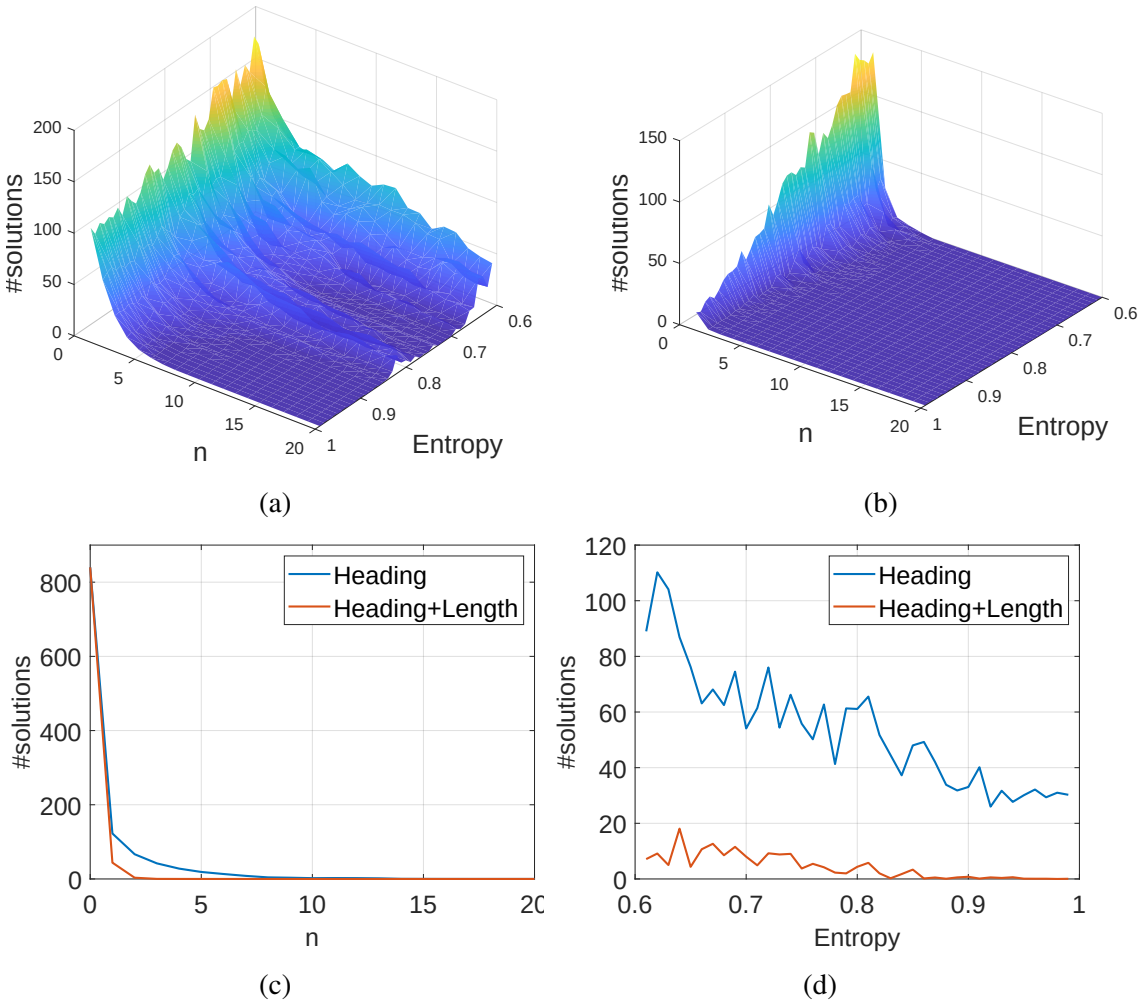


Figure 4.10: (a) #solutions with respect to map entropy values (heading only) and n. (b) #solutions with respect to map entropy values (heading+length) and n. (c) n versus #solutions with fixed map entropy = 0.86. (d) Map entropy values versus #solutions with $n = 3$. Adapted with permission from [2] © 2020 IEEE.

4.4.1.3 Physical Experiments

We also compare the two aforementioned methods in physical experiments. Again, the speed is described in n needed to reach a unique solution. Smaller n is more desirable. We test three sequences from CSData on CSMap, two sequences on KITTI00Map and two sequences on KITTI05Map. The comparison results are shown in the last two

columns of Tab. 4.1. In all tests, GBPL takes $n = 3.1$ in average with a standard deviation of 0.69 to localize the vehicle while PLAM takes $n = 6.3$ on average with a standard deviation of 2.29 in comparison. As expected, GBPL has a faster localization speed than that of PLAM. As shown in Tab. 4.1, the entropy values (heading+length) of CSMap, KITTI00Map and KITTI05Map are 0.724, 0.877, and 0.797, respectively. By checking the results in Fig. 4.10(b), n required for reaching a unique solution in the real map agrees with simulation results.

4.4.2 Localization Alignment and Verification Test

Global localization only provides an initial position and the accuracy of continuous localization is determined by the LAV thread. We show localization accuracy result for all seven test sequences. PLAM does not have the capability of continuous localization and hence is not tested here. We only compare GBPL result with the ground truth.

4.4.2.1 Ground Truth and Evaluation Metric

The ground truth in our experiments is the actual GPS trajectory. The localization error is defined as the Euclidean distance between the estimated aligned trajectory and the ground truth. The localization errors are measured in meters.

4.4.2.2 Accuracy Results

Figs. 4.11 and 4.12 show the accuracy results by plotting the localization errors of each sequence. Red vertical lines are where LAV is excuted, i.e., when turns are detected. The first red vertical line corresponds to where we obtain global location. In all test sequences, the error in vehicle position is reduced to less than $5m$ when LAV runs at the moments indicated by the red lines. After that error slowly grows until reaching the next LAV moment. This matches the expected map uncertainty (around $10m$). The localization accuracy of CSData on CSMap appears to be less than that of KITTI data. This is mostly

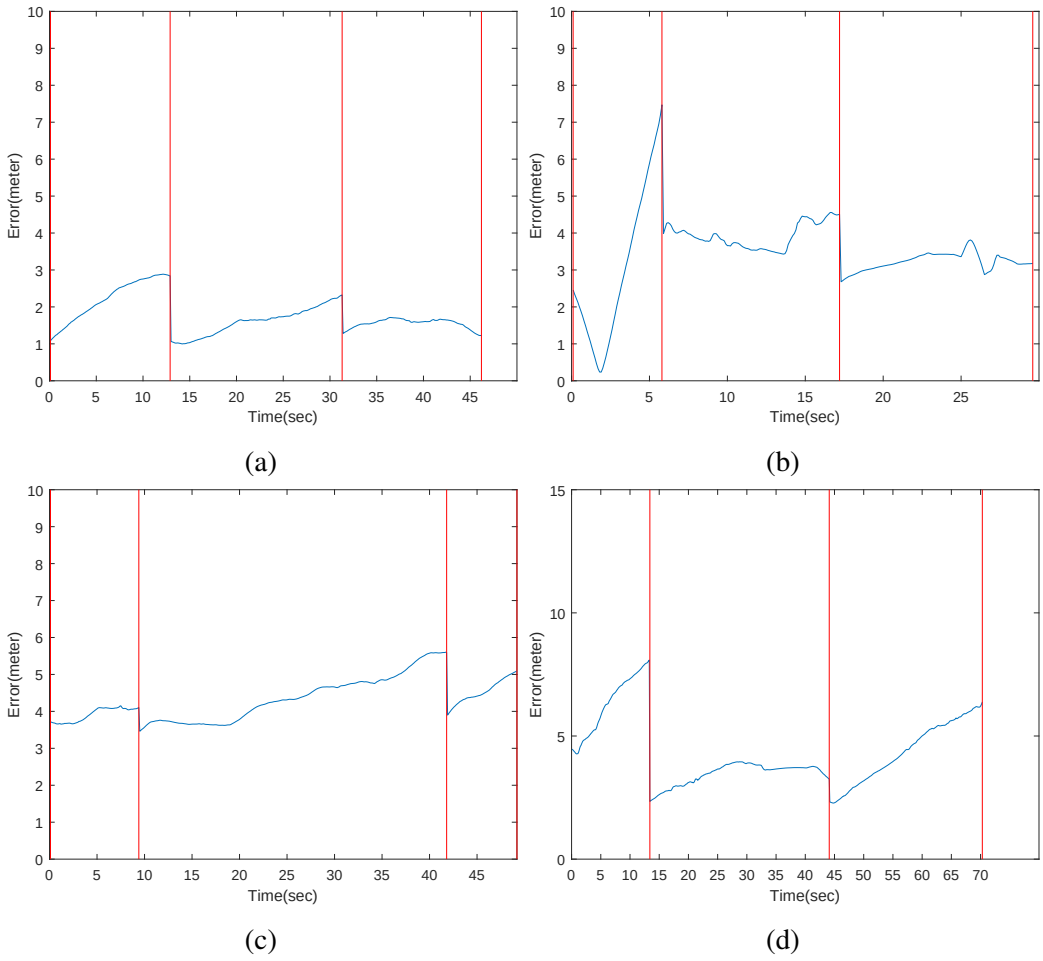


Figure 4.11: LAV accuracy results using KITTI sequences on KITTI00Map and KITTI05Map: (a) KITTI00-1, (b) KITTI00-2, (c) KITTI05-1, and (d) KITTI05-2. Adapted with permission from [2] © 2020 IEEE.

due to the fact that the ground truth of CSData is not as accurate as that of the KITTI dataset. CSData uses the GPS receiver on the cell phone with an accuracy of about 10 meters or worse while the GPS receiver for KITTI data set is high quality GPS (model RT3000v3) with an accuracy of 1 centimeter.

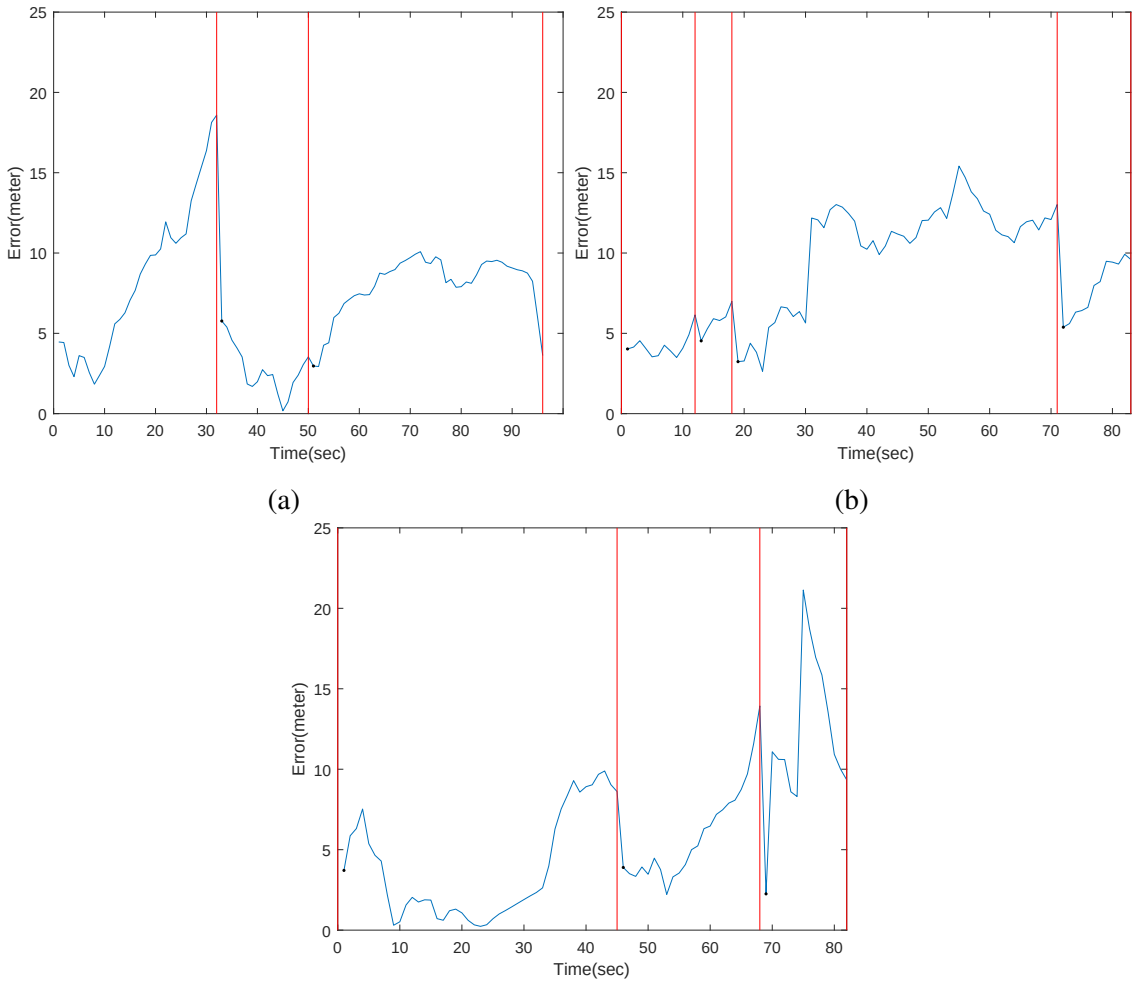
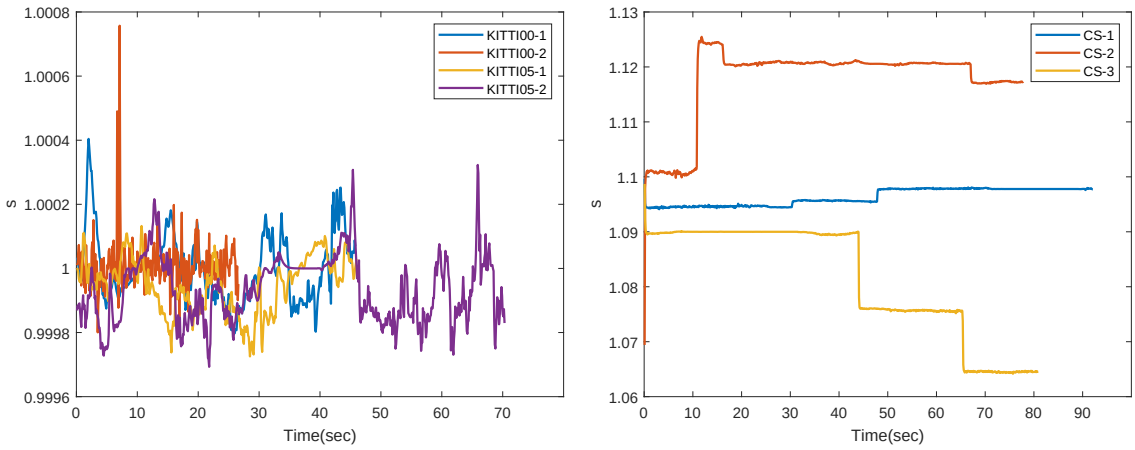


Figure 4.12: LAV accuracy results using CSData on CSMap: (a) CS-1, (b) CS-2, and (c) CS-3. Adapted with permission from [2] © 2020 IEEE.

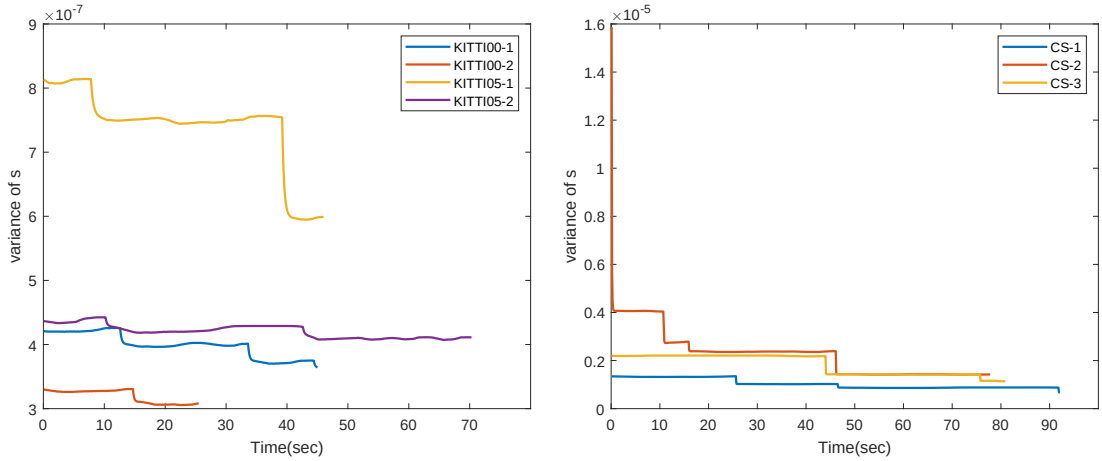
4.4.2.3 Scale and Slip Factor

Fig. 4.13 shows the estimated SSF in EKF (i.e. s_j in (4.10)). These results show the effectiveness of LAV in detecting systematic bias in wheel odometry. For CSData, SSF values are between 1.09 to 1.15 while the SSF values from KITTI data are close to 1.00. It is clear that the vehicle velocity from the Panda OBD II dongle contains bias. It tends to underestimated vehicle velocity by about 10%. This may be due to incorrect parameters



(a) (b)

Figure 4.13: Scale and slip factor value over time in EKF (4.10): (a) KITTI data and (b) CSData. Note the sequences are color coded and are not of the same length in time. Adapted with permission from [2] © 2020 IEEE.



(a) (b)

Figure 4.14: Scale and slip factor variance over time in EKF: (a) KITTI data and (b) CSData. Note the sequences are color coded and are not of the same length in time. Adapted with permission from [2] © 2020 IEEE.

in gear ratio or wheel/tire size. Also, the fluctuation in SSF in CSData is also large. This may also be a result of less accurate GPS values or variable tire inflation status since data

is collected at different times over several months. Nonrigid mounting of the cellphone also contributes to the issue. Nevertheless, our GBPL algorithm is robust to these factors and still provides a good localization result. We also shows the variance of s_j in Fig. 4.14. These results show $\sigma^2_{s_{ssf}}$ decreasing as travel length increases as in Lemma (4).

4.5 Conclusion

We reported our GBPL method that did not rely on the perception and recognition of external landmarks to localize robots/vehicles in urban environments. The proposed method is designed to be a fallback solution when everything else fails due to poor lighting conditions or bad weather conditions. The method estimated a rudimentry vehicle trajectory computed from an IMU, a compass, and a wheel encoder and matched it with a prior road map. To address the drifting issue in the dead-reckoning process and the fact that the vehicle trajectory may not overlap with road waypoints on the map, we developed a feature-based Bayesian graph matching where features are long and straight road segments. GBPL pre-processed maps into an HLG which stores all long and straight segments of road as nodes to facilitate global localization process. Once the map matching is successful, our algorithm tracks vehicle movement and use the map information to regulate EKF's drifting issue. The algorithm was tested in both simulation and physical experiments and results are satisfying.

5. VEHICLE-TO-VEHICLE COLLABORATIVE GRAPH-BASED PROPRIOCEPTIVE LOCALIZATION*

In Chapter 3 and Chapter 4, we propose two PL methods, PLAM and GBPL, as fall-back solutions that are naturally immune to these extreme environmental conditions because the PL methods only depend on a prior map and proprioceptive sensors such as inertial measurement units (IMUs) and/or wheel encoders. PL methods localize vehicles by extracting features from the estimated vehicle trajectories using the proprioceptive sensors and matching the features with a prior map.

However, the existing approaches suffer from strong dependence on trajectory types and slow convergence. Combining the PL method with modern vehicle-to-vehicle (V2V) communication which allows real time information exchange between vehicles, we design a new collaborative graph-based proprioceptive localization (C-GBPL) method (see Fig. 5.1). We extract trajectory features which are straight segments of trajectories and generate a merged query graph by combining inputs from neighboring vehicles. The localization problem becomes a graph-to-graph matching problem. Our algorithm outputs potential vehicle locations based on the the maximization of belief functions which often quickly converges to actual location over time.

Building our existing work on the single vehicle PL [2], our new collaborative framework alleviates the trajectory-dependence issue by exploiting the shared information. To facilitate the matching, we also pre-process the prior map which extends our heading-length graph (HLG) by adding super-vertices based on three different vehicle rendezvous types. The new algorithm that builds on new graph structure speeds up the matching com-

*Reprinted with permission from "Vehicle-to-Vehicle Collaborative Graph-based Proprioceptive Localization" by Hsin-Min Cheng, Chieh Chou and Dezhen Song. in *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 990-997, April 2021, Copyright © 2021 IEEE.

putation. Furthermore we explicitly prove that it can accelerate the convergence of belief functions for vehicle location. We have implemented the C-GBPL algorithm and tested it against the existing approach [2]. The C-GBPL algorithm significantly outperforms its single-vehicle counterpart in localization speed and is less sensitive to trajectory and map limitations.

5.1 Related Work

Our C-GBPL method is related to localization using different sensor modalities, map-based localization, and multi-robot localization.

Localization methods can be grouped into two categories: exteroceptive sensors and proprioceptive sensors according to sensor modalities. Exteroceptive sensors perceive external signals and recognize landmarks in the environment to estimate location. Mainstream exteroceptive sensors include cameras [14,19,20], laser range finders [26], and GPS receivers [27,29]. These methods are considerably susceptible to adversary environmental conditions such as low visibility, extreme weather conditions, or electromagnetic inference. On the contrary, proprioceptive sensors, such as IMUs [4] and wheel encoders [31] do not rely on external signals and are inherently immune to external conditions. However, they are more susceptible to error drift. Recent approaches combining a camera or a laser ranger finder with an IMU [5] which design is an exteroceptive-proprioceptive sensor fusion approach, greatly improves system robustness and has become popular in applications. However, these approaches still heavily rely on their exteroceptive sensors and cannot function properly under the aforementioned adversary environmental conditions.

Our PL method is a map-based localization [19,40–43]. Thrun et al. [12] classify map representation into two categories: the location-based and the feature-based. The location-based maps are represented with specific locations of objects. For example, those existing geographic maps consisted of coordinate of locations such as Google Maps™ [45]

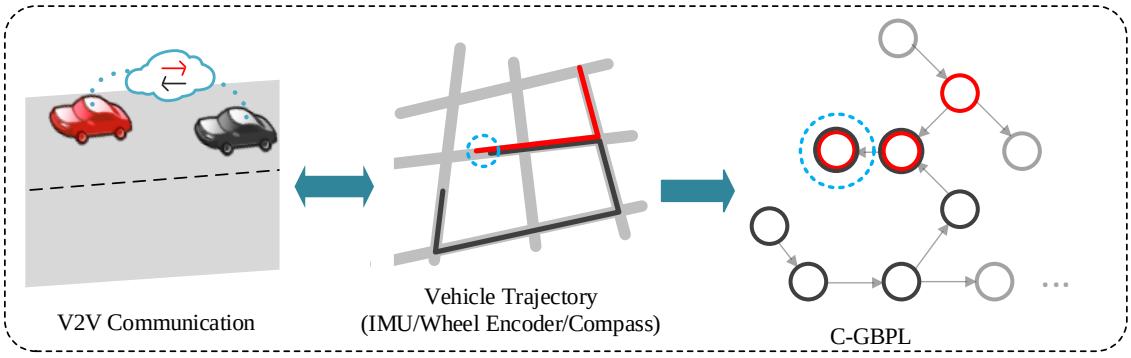


Figure 5.1: An illustration of C-GBPL method using a two-vehicle rendezvous case. The ego vehicle is colored in red and the other vehicle is in black. At rendezvous, we aggregate vehicle trajectories to form a merged feature graph to facilitate localization. Reprinted with permission from [3] © 2021 IEEE.

and OpenStreetMaps™ (OSM) [44]. Geographic maps often based on GPS measurements. Researchers develop map matching techniques such as point-to-point, point-to-curve, curve-to-curve matching, or advanced extensions [46]. The feature-based map consists of features of interest at its location. An example is ORB features [48] for visual simultaneous localization and mapping. Our PL methods extract heading-length features from proprioceptive sensors and prior maps to convert a location-based map matching to a feature-based map matching which improves localization robustness to sensor drift and also speeds up computation in the process.

In the area of multi-robot research, decentralized estimation of robot poses has gained considerable attention [71–74]. The multi-robot systems outperform single-robot systems in many aspects, such as improving localization efficiently, reducing computational cost, increasing accuracy and fault tolerance, and accelerating map exploration and coverage. Our work is related to multi-robot localization in particular [71, 75, 76]. To tackle the decentralized multi-robot localization problem, researchers propose [75] and use [74] the concept of checkpoints representing delayed synchronization of observation after ex-

changes of information between robots. Also, the concept is extended to the decentralized information transfer scheme [77] based on communication constraints. Our problem is similar in the way that we can benefit from exchanged motion information from other robots, but it is different because we do not rely on landmarks or exteroceptive sensing to acquire relative positions with other robots.

PL methods are gaining attention in vehicle localization [1, 2, 40, 66, 67], all of which are map-aided localization using proprioceptive sensors. Wahlstrom et al. [66] employ a minimal setup using vehicle speed and speed limit information map. Yu et al. [67] develop an extended Kalman filter (EKF)-based dead reckoning approach based on odometer and gyroscope readings and a map is used to correct errors. However, an initial position from GPS is required for both methods. Our localization solution does not require a known initial position. In [40], the velocity from the wheel encoder and steering angle are used for odometry and a particle filter based map matching scheme helps estimating vehicle positions.

Our group studies localization using proprioceptive sensors under different setups [1, 2]. This paper extends our prior work [2] for a single vehicle to a multiple-vehicle PL method.

5.2 Problem Formulation and System Design

All vehicles have a prior map of the city. Each vehicle is equipped with proprioceptive sensors including an IMU and an on-board diagnostics (OBD) scanner which provides velocity feedback (similar to a wheel encoder). To compensate for direction drift, each vehicle has a digital compass. To formulate our collaborative localization problem, we have the following assumptions:

a.0 The ego vehicle is able to navigate in the environment and make turns at appropriate locations. All vehicles are nonholonomic.

a.1 The prior road map contains straight segments in most part of its streets and streets are not strict grids with equal side lengths.

a.2 IMU and compass are co-located, pre-calibrated, and fixed inside the vehicle. Their readings are synchronized and time-stamped.

a.3 Vehicles communicate with each other in close range and we can assume that they are on the same street or same intersection.

As part of the input of the problem, a prior road map consisting of a set of roads with GPS waypoints is required. Common notations are defined as follows,

- $\mathcal{M}_p$ represents the prior road map which is a set of GPS positions.
- $\mathbf{z} = \{\mathbf{a}, \omega, \phi, \mathbf{v}\}$ denote $in\ situ$ sensory readings of vehicle, where $\mathbf{a}$ denotes accelerometer readings of the IMU, ω denotes gyroscope readings of the IMU, ϕ denotes compass readings, and $\mathbf{v}$ denotes velocity readings.

The C-GBPL problem is defined as follows.

Problem 3. *When ego vehicle l rendezvous with vehicle l', localize vehicles collaboratively given sensory reading $\mathbf{z}$, $\mathbf{z}'$, and $\mathcal{M}_p$.*

It is worth mentioning that this problem formulation only concerns a two-vehicle rendezvous case. However, it is the atomic case of multiple vehicles rendezvous case because an n-vehicle rendezvous case can be easily decomposed into a sequence of $n-1$ two-vehicle rendezvous cases.

5.3 Algorithm

Since our algorithm builds on GBPL algorithm, we begin with a brief review of GBPL algorithm [2] in Section 5.3.1. Then we will extended it into C-GBPL in Section 5.3.2.

5.3.1 GBPL Review

As a single vehicle localization method, GBPL employs the proprioceptive sensors to estimate vehicle trajectory and match it with a prior map. GBPL is a feature-based map matching method instead of raw trajectory matching because 1) there is a significant drift issue in the dead reckoning process and 2) the vehicle trajectory may not match the GPS waypoints on the map where a street may contain multiple lanes and trajectories may differ due to lane selection or different traffic patterns. GBPL has three main building blocks: *heading length graph (HLG) construction, query generation from sensory data, and vehicle localization*. GBPL can be viewed as a feature-based map matching with each feature to be a straight segment of a road with heading and length as its feature descriptors. *HLG construction* reduces the prior map $\mathcal{M}_p$, which is a set of GPS waypoints, to a discrete feature structure HLG $\mathcal{M}_h$ to facilitate the feature matching. *Query generation* extracts features from dead reckoning trajectory based on the proprioceptive sensors. The *vehicle localization* performs the feature-based Bayesian map matching and vehicle tracking afterwards.

5.3.1.1 HLG construction

HLG is a feature representation of the prior map. In HLG, a vertex $v_i \in \mathcal{V}_h$ represents a straight and continuous road segment with no intersections. An edge $e_{i,i'} \in \mathcal{E}_h$ characterizes the orientation change between the connected two vertices v_i and $v_{i'}$. $\mathcal{M}_h$ have two types of edges: road intersections and curve segments; and two types of vertices: long straight segment vertices and short transitional segment vertices. The long straight segment vertices are used for heading-length matching later. The short transitional segment vertices are often formed between curve segments or curved roads entering intersections.

Each vertex contains the following information

$$v_i = \{\mathbf{X}_i, \theta_i, d_i, b_i\}, \tag{5.1}$$

where $\mathbf{X}_i = [\mathbf{x}_{i,s}^\mathsf{T}, \cdots, \mathbf{x}_{i,e}^\mathsf{T}]^\mathsf{T}$ contain all 2D positions of waypoints on the road segment, orientation $\theta_i \in (-\pi, \pi]$ is the angle between the geographic north, d_i is road segment length, and b_i is the binary variable indicate if the vertex is a long road segment. Only long road segments ($b_i = 1$) are used in localization which defines vertex subset $\mathcal{V}_{h,l} \subset \mathcal{V}_h$ corresponding to long straight segments.

5.3.1.2 Query Generation

When the vehicle is driving down the road, we can estimate the trajectory from sensory readings with an EKF-based approach and generate a query heading-length sequence. In the state representation, the state vector $\mathbf{X}_{s,j}$ at time j of the EKF is:

$$\mathbf{X}_{s,j} := [\mathbf{p}_j^I, \mathbf{v}_j^I, \mathbf{\Theta}_j^I]^\mathsf{T},$$

where $\mathbf{p}^I \in \mathbb{R}^3$, velocity $\mathbf{v}^I \in \mathbb{R}^3$, and the X-Y-Z Euler angles $\mathbf{\Theta}^I$ in fixed inertial frame $\{I\}$. The EKF-based dead reckoning provides a vehicle trajectory but is inevitably drift-prone. Instead of using it for directly matching to the prior map, we extract heading and length of the straight segments for our feature-based matching. The resulting query heading-length sequence is denoted by

$$Q := \{\Theta_q, D_q\},$$

where $\Theta_q = \{\Theta_{q,k} | k = 1, \cdots, n\}$, $D_q = \{d_{q,k} | k = 1, \cdots, n\}$, and $\Theta_{q,k}$ and $d_{q,k}$ are the observations of heading and length from EKF for a straight segment k.

5.3.1.3 *Vehicle localization*

This is two-step process: i) perform global graph match that finds the query-HLG match and ii) track the vehicle location to provide continuous localization result after a global match is identified. Since the second step is the trivial, we focus on global graph match only. Given $Q = \{\Theta_q, D_q\}$, let us denote a candidate heading-length vertex sequence in $\mathcal{M}_h$ by $M := \{\Theta, D\} = \{\{\theta_k, d_k\}|k = 1, \cdots, n\}$ correspondingly. As a convention in this paper, for random vector $\star$, $\mu_\star$ represents its mean vector. The belief that Q and M match is the following conditional probability

$$P(\mu_Q = \mu_M|Q, M) \propto \prod_{k=1}^{n} f_T(t(\theta_{q,k}, \theta_k)) f(z(d_{q,k}, d_k)) \tag{5.2}$$

due to independent sensor noises where $f_T(t(\theta_{q,k}, \theta_k))$ is the probability density function (PDF) of Student's t-distribution and $f(\cdot)$ is the PDF of standard normal distribution. As the length of the matching sequence grows, the belief function converges for the correct matching and a global location is identified when thresholding condition is satisfied and only one solution remains. We search for the best matching by generating different candidate sequences on the map using breadth-first search on the HLG.

5.3.2 *C-GBPL*

In a single vehicle case, we simply match query sequence with a candidate sequence constructed from the HLG. This changes when vehicles can talk to each other. The *rendezvous events* describe moments when a vehicle moves into another vehicle's communication range (assumption a.3). At the moment of rendezvous, one vehicle can pass its query sequence to the other. The two query sequences combine into a query graph which will be matched against HLG. The matching problem evolves into a graph-to-graph matching problem.

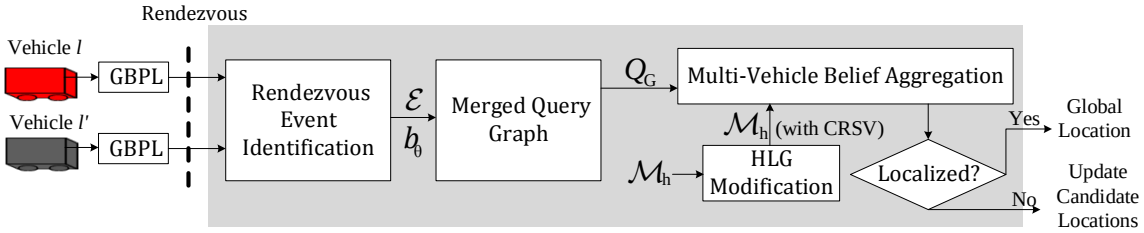


Figure 5.2: C-GBPL system diagram. Reprinted with permission from [3] © 2021 IEEE..

Fig. 5.2 show the system diagram of C-GBPL. Before rendezvous, each vehicle runs GBPL algorithm in [2]. In C-GPBL, we propose the following building blocks to compose and solve the graph-to-graph matching problem to simultaneously improve localization efficiency and reduce computational cost. The first block is *HLG modification* where we modify HLG with super vertex groups to capture potential rendezvous locations to facilitate the graph-to-graph matching process. With the modified HLG, the remaining three blocks are *rendezvous event identification*, *merged query graph*, and *multi-vehicle belief aggregation*. We will detail each block in separate subsections.

5.3.2.1 HLG Modification

At rendezvous, the vehicle pair must be within communication range of each other to exchange information (Assumption a.3). This only occurs with a limited set of possibilities and can be utilized to facilitate graph-matching because we can trim the searching space on subset of vertices in $\mathcal{M}_h$ by focusing on the possible rendezvous locations. We capture all possible rendezvous locations by augmenting the pre-processed HLG with an additional layer which are candidate rendezvous super vertex (CRSV) groups (see Fig. 5.3) which only have three types.

- Type 0: *same vertex*. This type is embedded in the original HLG and does not require additional processing. The number of Type 0 CRSV is $O(\mathcal{V}_{h,l})$.

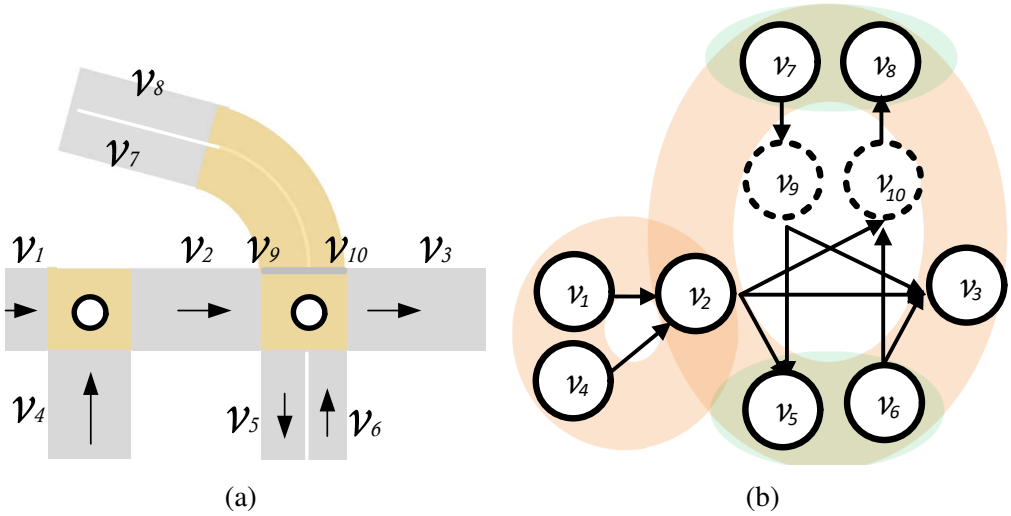


Figure 5.3: HLG Modification. (a) A prior road map overlay with edges (in light yellow) and vertices (in gray). (b) An illustration of two CRSV types: Type 1 coupled vertices (green color) and Type 2 intersection sharing vertices (orange color). Reprinted with permission from [3] © 2021 IEEE.

- Type 1: *coupled vertices.* We define coupled vertices by pair of vertices with opposite directions on same road segment. In Fig. 5.3(b), we show examples of coupled vertices in green color. The number of Type 1 CRSV is also $O(\mathcal{V}_{h,l})$.

- Type 2: *intersection sharing vertices.* We define intersection sharing vertices as a group of vertices sharing the same road intersection. For vertices in a group, the corresponding road segments share the same intersection. In some cases that the road intersections are connected with a curved road segment (e.g. an edge in $\mathcal{M}_h$), the nearest vertices are included instead. We show an example in Fig. 5.3(b) where v_7 and v_8 are considered as intersection sharing vertices. The number of Type 2 CRSV is $O(\mathcal{V}_{h,l})$ which happens when a map consists with square grids and bi-directional roads.

5.3.2.2 *Rendezvous Event Identification*

On the vehicle side, we also need to identify corresponding rendezvous types using on-board sensors.

Recall that the ego-vehicle index is denoted by l and the other vehicle index by l', where $l, l' \in L$. We develop algorithm from ego-vehicle l view and as it is established vice versa for the other vehicle l' where prime symbol $'$ indicates the other vehicle. We label the vehicle status by 'V' or 'E' based on whether it is on a vertex or an edge. For vehicles (l, l'), this results in four kinds of rendezvous events denoted by $\mathcal{E}_{E,E}$, $\mathcal{E}_{E,V}$, $\mathcal{E}_{V,E}$, and $\mathcal{E}_{V,V}$. To further reduce the four kinds into the three types in the CRSV groups, we check whether vehicles (l, l') have the same headings at rendezvous. Note whenever vehicle changes heading, it does not have stable heading. Thus we specify vehicle rendezvous heading by its *latest stable* heading. We set binary variable to $b_\theta = 1$ if vehicle (l, l') have same headings and $b_\theta = 0$ otherwise. See Fig. 5.4 for examples. The three types are identified as follows,

- Type 0: When $b_\theta = 1$, vehicles travel through same vertex in all four rendezvous events as shown in top four figures in Fig. 5.4. Vehicle trajectories are linked by hidden super vertices (Type 0) of HLG.

- Type 1: When $b_\theta = 0$, $\mathcal{E}_{V,V}$ and vehicles have opposite orientations, vehicles travel through vertices corresponding to same road segment with opposite directions. Their trajectories are linked by coupled vertices (Type 1) of HLG. This case is shown in the right bottom of Fig. 5.4.

- Type 2: When $b_\theta = 0$ and $\{\mathcal{E}_{E,E}, \mathcal{E}_{E,V}, \mathcal{E}_{V,E}\}$ or $\mathcal{E}_{V,V}$ and vehicles have different orientations other than opposite, vehicles rendezvous at intersection from different directions which are connected by intersection sharing vertices (Type 2) of HLG.

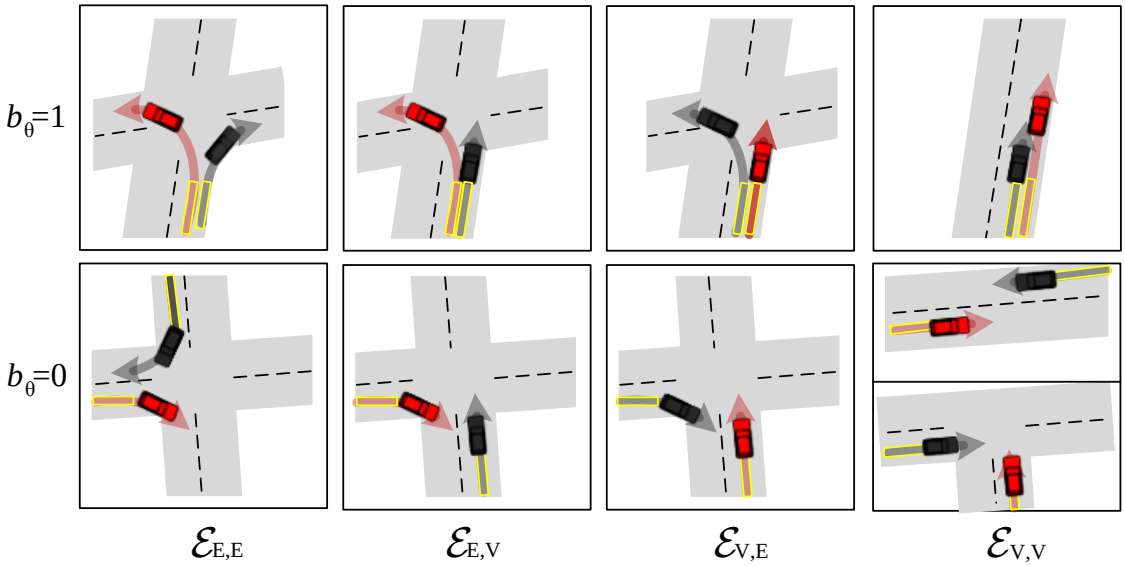


Figure 5.4: Rendezvous events with same/different heading (b_θ) which we use to identify vehicle relative locations. The yellow rectangular boxes mark out the latest vehicle stable headings used to determine b_θ. Reprinted with permission from [3] © 2021 IEEE.

5.3.2.3 Merged Query Graph

With different rendezvous event types identified, we can combine individual query sequences and form a merged query graph. We denote the merged query graph by Q_G which consists of nodes and edges in query sequences/graph Q and Q' for vehicles l and l', respectively. They are connected together using the type info.

5.3.2.4 Multi-Vehicle Belief Aggregation

The main part of global localization is the matching of the merged query graph Q_G to the corresponding part on the modified HLG. This requires us to establish a belief function to evaluate Q_G and a candidate matching graph M_G on the map.

The type associated with Q_G helps us identify the corresponding M_G which consist of nodes in M and M', where M and M' are candidate sequences for vehicles l and l',

respectively.

At rendezvous, the matching belief from vehicle l' is aggregated with matching belief from ego-vehicle l. We extend the single vehicle belief function in (5.2) to a multi-vehicle belief.

Lemma 5. *The multi-vehicle belief function is the conditional probability for the joint belief function which is obtained by multiplying individual components below.*

$$\begin{aligned} &P(\mu_Q = \mu_M, \mu_{Q'} = \mu_{M'}|Q, M, Q', M') \\ &\quad = P(\mu_Q = \mu_M|Q, M)P(\mu_{Q'} = \mu_{M'}|Q', M') \\ &\quad \propto \prod_{k=1}^{n} f_T(t(\theta_{q,k}, \theta_k))f(z(d_{q,k}, d_k))\times \\ &\qquad \prod_{k'=1}^{n'} f_T(t(\theta_{q,k'}, \theta_{k'}))f(z(d_{q,k'}, d_{k'})), \end{aligned} \tag{5.3}$$

where n and n' are number of observations in Q and Q', respectively, and k and k' are index variables.

Proof. Note that $\mu_Q = \mu_M$ and $\mu_{Q'} = \mu_{M'}$ are independent given Q, M, Q' and M'. We decompose (5.3) into two terms

$$\begin{aligned} &P(\mu_Q = \mu_M, \mu_{Q'} = \mu_{M'}|Q, M, Q', M') \\ &=P(\mu_Q = \mu_M|Q, M, Q', M')P(\mu_{Q'} = \mu_{M'}|Q, M, Q', M') \end{aligned} \tag{5.4}$$

Since $\mu_Q = \mu_M$ is independent of Q', M' given Q, M, we write

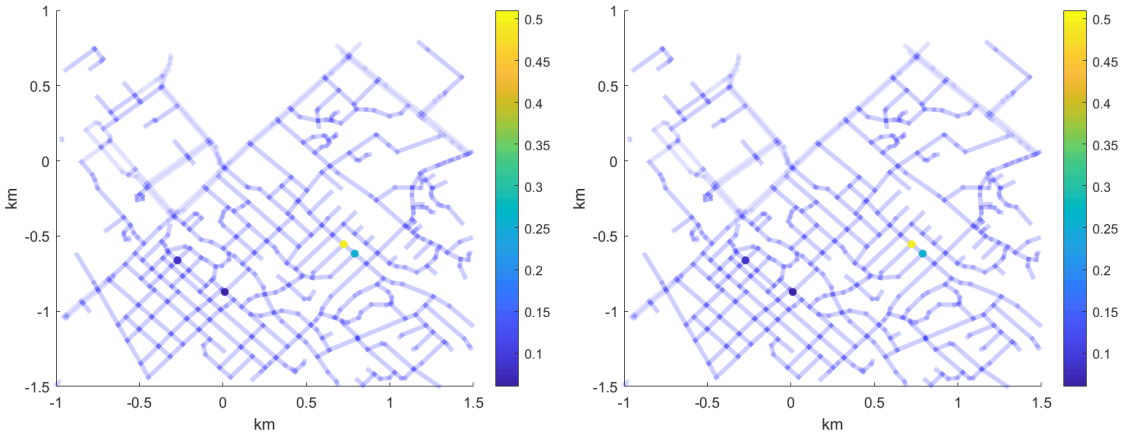
$$P(\mu_Q = \mu_M|Q, M, Q', M') = P(\mu_Q = \mu_M|Q, M), \tag{5.5}$$

which is the belief that Q and M match in (5.2). Similarly,

$$P(\mu_{Q'} = \mu_{M'}|Q, M, Q', M') = P(\mu_{Q'} = \mu_{M'}|Q', M'). \tag{5.6}$$

Plugging (5.5), (5.6) using the form derived in (5.2) into (5.4), we obtain the lemma. □

Fig. 5.5 illustrates the ego vehicle belief and aggregated multi-vehicle belief after rendezvous with the other vehicle.



(a) (b)

Figure 5.5: An example of multi-vehicle belief aggregation (best viewed in color). The probability value of each location is colored according to color bar. (a) The candidate locations with corresponding probabilities of ego vehicle. (b) After rendezvous with the other vehicle, the candidate locations and probabilities are updated. There is a significant drop in number of candidate locations after rendezvous. Reprinted with permission from [3] © 2021 IEEE.

5.3.2.5 Algorithm Framework

Lemma 5 provides us with a method to localize the vehicle by thresholding the belief function over candidate solution M_G. It can be done by applying a breadth-first search

strategy starting with matching CRSV types. We summarize the C-GBPL framework in Algorithm 2. We denote the set of Type 1 CRSV by $\mathcal{V}_{\mathrm{T}_1}$ where $\mathcal{V}_{\mathrm{T}_1} = \{V_{\mathrm{T}_{1,m_1}} | m_1 = 1, \cdots, n_{\mathrm{T}_1}\}$ and each $V_{\mathrm{T}_{1,m_1}}$ contain vertices in the same group. n_{T_1} is the cardinality of $\mathcal{V}_{\mathrm{T}_1}$. Similarly, We define $\mathcal{V}_{\mathrm{T}_2} = \{V_{\mathrm{T}_{2,m_2}} | m_2 = 1, \cdots, n_{\mathrm{T}_2}\}$. Recall that $\mathcal{V}_{h,l}$ is vertex set of HLG. For $v_i \in \mathcal{V}_{h,l}$, we augment information of CRSV type and element index for searching purpose. Note that v_i may belong to more than one CRSV types.

Not all pairs of candidate sequences M and M' satisfy the CRSV type constraint. Let us use Fig. 5.6 as an example. In the left figure, we show trajectories of two vehicles and their rendezvous. The right figure shows the graph matching between M and M' in HLG $\mathcal{M}_h$. In this example, the Q_G is classified as Type 2 and thus M_G belongs to Type 2 CRSV group. We show three candidates (M'_1, M'_2, M'_3) for ego vehicle. Only M'_1 satisfies the CRSV and feature sequence matching constraints and both M'_2 and M'_3 are trimmed.

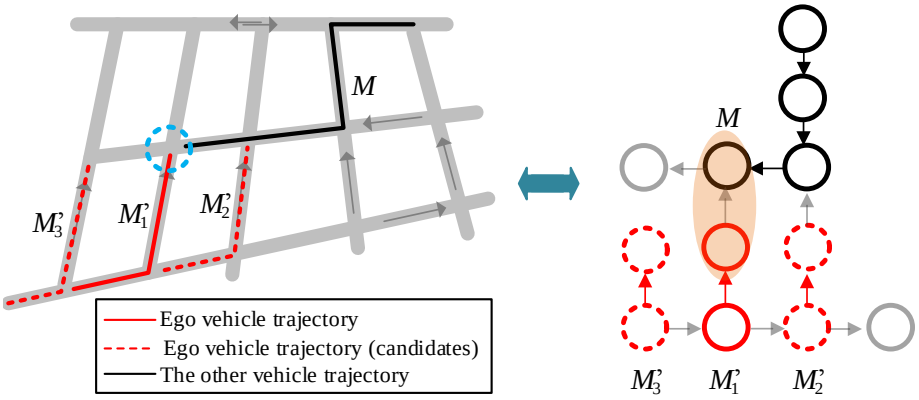


Figure 5.6: An example of graph-based candidate trimming. Left figure: we show trajectories of two vehicles and their rendezvous. Right figure: HLG $\mathcal{M}_h$ and candidate heading-length vertex sequence M and M'. We show three candidates (M'_1, M'_2, M'_3) for ego vehicle and trim candidates are not correct CRSV type, both M'_2 and M'_3 are trimmed. Reprinted with permission from [3] © 2021 IEEE.

To reduce possible prior correlations, we use the following two rules. First, when

the ego vehicle rendezvouses the other vehicle, we do not update its belief if they have same candidate vertices. Second, we only update the ego vehicle belief once for the same rendezvous with the other vehicle. According to assumption a.3, this implies the next rendezvous happens when either vehicles has new observations.

In Lemma 5, the product format of belief function in (5.2) still holds for multi-vehicle case as described in (5.5) and (5.6). The only difference is the number of nodes involved in the product. Computing the multi-vehicle beliefs in (5.3) can be computation intensive if we do not effectively reuse the prior computation. We store the prior computation from each individual vehicle and exchange the information between vehicles. For each vehicle, we define the candidate vertex information set $\mathcal{C}_k$ where $k = 1, \cdots, n$ is the length of the query sequence. The candidate vertex set is denoted by $\mathcal{C}_k = \{\{v_i, \mathcal{V}_{M,i}, p_i\} | i = 1, \cdots, n_{\mathcal{C}_k}\}$, where each element in $\mathcal{C}_k$ record the candidate vertex v_i (the starting vertex of the trajectory/path), $\mathcal{V}_{M,i}$ is the set of vertex path, and the matching probability p_i in (5.2) and $n_{\mathcal{C}_k}$ is the cardinality of $\mathcal{C}_k$. By thresholding the belief function over candidate solutions, we might still get many candidate solutions because the hypothesis is conservative in rejection as in GBPL [2]. The Otsu method [63] can be applied to further trim candidate trajectories with lower probabilities. If more than one solution survives, it indicates that more observations are needed to localize the vehicle.

We now analyze the complexity of C-GBPL algorithm. The upper bound of candidate vertex cardinality is $\mathcal{V}_{h,l}$ and thus it takes $O(|\mathcal{V}_{h,l}|)$ to traverse $\mathcal{C}_n$ and $\mathcal{C}'_n$. For Type 0 cases, we can find set intersection of $\mathcal{V}_i$ and $\mathcal{V}'_i$ in $O(|\mathcal{V}_{h,l}| \log(|\mathcal{V}_{h,l}|))$ by sorting and binary search. Similarly, for Type 1 or Type 2 cases, we can find (v_i, v'_i) pairs with the same group index in $O(n_{\mathrm{T}_1} \log(n_{\mathrm{T}_1}))$ or $O(n_{\mathrm{T}_2} \log(n_{\mathrm{T}_2}))$. In the worst case scenario, both n_{T_1} and n_{T_2} are $O(|\mathcal{V}_{h,l}|)$. Then line 7, 10, 13 are the same in complexity of $O(|\mathcal{V}_{h,l}| \log(|\mathcal{V}_{h,l}|))$. The classification of probabilities into two groups is $O(|\mathcal{V}_{h,l}|)$ using Hoare's selection algorithm. We summarize the computational complexity of Algorithm 2 in Lemma 6.

Lemma 6. *The computation complexity of the C-GBPL is* $O(|\mathcal{V}_{h,l}|\log(|\mathcal{V}_{h,l}|))$.

5.3.3 Localization Analysis

Intuitively, combining inputs from the other vehicle helps the ego-vehicle to reduce ambiguity in map matching and hence leads to faster convergence in the localization process. Let us show this by analyzing the how conditional probability of localization given the matching sequence changes.

Let us define three binary events: $A_k = 1$ if $\mu_{d_{q,k}} = \mu_{d_k}$, $B_k = 1$ if $\mu_{\theta_{q,k}} = \mu_{\theta_k}$, and $C_k = 1$ if vertex k in M_h is the actual location. Same as [2], we employ hypothesis testing to reject unlikely matching by setting the significance level α, where α is a small probability. According to the definition, $P(A_k|C_k) = (1-\alpha)$ and and $P(B_k|C_k) = (1-\alpha)$ are the conditional probabilities that a correct matched sequence survive the test for pair $\mu_{d_{q,k}} = \mu_{d_k}$ and $\mu_{\theta_{q,k}} = \mu_{\theta_k}$ correspondingly. For convenience, for vehicle l we define joint events: $\mathcal{A} = A_1 \cdots A_n$, $\mathcal{B} = B_1 \cdots B_n$, and $\mathcal{C} = C_1 \cdots C_n$. The joint event $\mathcal{C}$ is equivalent to say that $M := \{\Theta, D\}$ represents the true trajectory, whereas we know joint event $\mathcal{AB}$ from sequence matching. Similarly, for vehicle l' we define joint events: $\mathcal{A}'$, $\mathcal{B}'$, and $\mathcal{C}'$. We define a binary event E if vehicle l rendezvous with l'. Also, there are n observations for vehicle l and n' observations for vehicle l' at the rendezvous.

In the analysis, we denote $n_v = |\mathcal{V}_{h,l}|$ as the cardinality of $\mathcal{V}_{h,l}$ and n_b as the expected number of neighbors for each vertex. n_b depends on how many streets an intersection has. We describe map/trajectory property in a rudimentary way by assuming k_d levels of distinguishable discrete headings in $[0, 2\pi)$ and k_l levels of distinguishable discrete road lengths. Each vertex takes a heading value and length value with equal probabilities of $1/k_d$ and $1/k_l$ correspondingly. Generally speaking, we know $n_v \gg k_d \geq n_b$ and $n_v \gg k_l \geq n_b$ for most maps. We denote n_c the total observations of vehicle l combining of vehicle l' trajectory given event E. We have the following lemma.

Algorithm 2: C-GBPL

Input: $\mathcal{M}_h$, $\{\mathcal{C}_n, Q\}$ and $\{\mathcal{C}'_n, Q'\}$
Output: vehicle l location or updated $\{\mathcal{C}_n, Q\}$

1	$\mathcal{C}_{res} \leftarrow \emptyset$, $p_{res} \leftarrow \emptyset$	$O(1)$
2	Identify rendezvous event and Recognize CRSV type using Q and Q'	$O(1)$
3	Form merged query graph Q_G	$O(1)$
4	$\mathcal{V}_i \leftarrow$ latest vertices in $\mathcal{C}_n$, $\mathcal{V}'_i \leftarrow$ latest vertices in $\mathcal{C}'_n$	$O(\lvert\mathcal{V}_{h,l}\rvert)$
5	**switch** *CRSV Type* **do**	
6	**case** *'Type 0'* **do**	
7	$\mathcal{V}_i \leftarrow$ intersect of $\mathcal{V}_i$, $\mathcal{V}'_i$	$O(\lvert\mathcal{V}_{h,l}\rvert \log(\lvert\mathcal{V}_{h,l}\rvert))$
8	**case** *'Type 1'* **do**	
9	$\mathcal{V}_i \leftarrow \forall v_i \in \mathcal{V}_i$ with label $\mathtt{T}_1$, $\mathcal{V}'_i \leftarrow \forall v'_i \in \mathcal{V}'_i$ with label $\mathtt{T}_1$	$O(\lvert\mathcal{V}_{h,l}\rvert)$
10	Find (v_i, v'_i) pair with same element index;	$O(n_{\mathtt{T}_1} \log(n_{\mathtt{T}_1}))$
11	**case** *'Type 2'* **do**	
12	$\mathcal{V}_i \leftarrow \forall v_i \in \mathcal{V}_i$ with label $\mathtt{T}_2$, $\mathcal{V}'_i \leftarrow \forall v'_i \in \mathcal{V}'_i$ with label $\mathtt{T}_2$	$O(\lvert\mathcal{V}_{h,l}\rvert)$
13	Find (v_i, v'_i) pair with same element index;	$O(n_{\mathtt{T}_2} \log((n_{\mathtt{T}_2}))$
14	**for** $v_i \in \mathcal{V}_i$ **do**	$O(\lvert\mathcal{V}_{h,l}\rvert)$
15	Access (v_i, v'_i) pair information	$O(1)$
16	$p \leftarrow p_i \cdot p'_i$	$O(1)$
17	$p_{res} \leftarrow p_{res} \cup p$	$O(1)$
18	$\mathcal{C}_{res} \leftarrow \mathcal{C}_{res} \cup \mathcal{C}_{n,i}$	$O(1)$
19	Classify p_{res} using Otsu's method;	$O(\lvert\mathcal{V}_{h,l}\rvert)$
20	Remove elements in $\mathcal{C}_{res}$ with lower probabilities;	$O(1)$
21	**if** $\lvert\mathcal{C}_{res}\rvert == 1$ **then**	
22	Return vehicle location;	$O(1)$
23	**else**	
24	$\mathcal{C}_n \leftarrow \mathcal{C}_{res}$	$O(1)$
25	Return $\{\mathcal{C}_n, Q\}$;	$O(1)$

Lemma 7. *The joint conditional probability that $M = \{\Theta, D\}$ is the true matching sequence given $Q = \{\Theta_q, D_q\}$ matches M, M' is the true matching sequence given Q' matches M' with rendezvous is,*

$$P(\mathcal{C}, \mathcal{C}' | \mathcal{A}, \mathcal{B}, \mathcal{A}', \mathcal{B}', E) = \left[(1-\alpha)^2 k_d k_l\right]^{n+n'} \frac{1}{n_b^{n_c-1}} \frac{1}{n_v}. \tag{5.7}$$

Proof. Applying the Bayesian equation, we have

$$\begin{aligned} & P(\mathcal{C}, \mathcal{C}' | \mathcal{A}, \mathcal{B}, \mathcal{A}', \mathcal{B}', E) \\ =& \frac{P(\mathcal{A}, \mathcal{B}, \mathcal{A}', \mathcal{B}', E | \mathcal{C}, \mathcal{C}') P(\mathcal{C}, \mathcal{C}')}{P(\mathcal{A}, \mathcal{B}, \mathcal{A}', \mathcal{B}', E)} \end{aligned} \tag{5.8}$$

Since $\mathcal{A}$, $\mathcal{B}$, $\mathcal{A}'$, $\mathcal{B}'$ are conditional independent to E given $\mathcal{C}$ and $\mathcal{C}'$, we have $P(\mathcal{A}, \mathcal{B}, \mathcal{A}', \mathcal{B}', E|\mathcal{C}, \mathcal{C}') = P(\mathcal{A}, \mathcal{B}, \mathcal{A}', \mathcal{B}'|\mathcal{C}, \mathcal{C}')P(E|\mathcal{C}, \mathcal{C}')$. Also, $\mathcal{A}$ and $\mathcal{B}$ are conditional independent to $\mathcal{A}'$ and $\mathcal{B}'$ given $\mathcal{C}$ and $\mathcal{C}'$ since each vehicle perform GBPL independently. Thus we have $P(\mathcal{A}, \mathcal{B}, \mathcal{A}', \mathcal{B}'|\mathcal{C}, \mathcal{C}') = P(\mathcal{A}, \mathcal{B}|C)$ and $P(\mathcal{A}, \mathcal{B}, \mathcal{A}', \mathcal{B}'|\mathcal{C}, \mathcal{C}') = P(\mathcal{A}', \mathcal{B}'|\mathcal{C}')$. We rewrite (5.8) by

$$\begin{aligned} & P(\mathcal{C}, \mathcal{C}'|\mathcal{A}, \mathcal{B}, \mathcal{A}', \mathcal{B}', E) \\ &= \frac{P(\mathcal{A}, \mathcal{B}|C)P(\mathcal{A}', \mathcal{B}'|\mathcal{C}')P(E|\mathcal{C}, \mathcal{C}')P(\mathcal{C}, \mathcal{C}')}{P(\mathcal{A})P(\mathcal{B})P(\mathcal{A}')P(\mathcal{B}')P(E)} \\ &= \frac{P(\mathcal{A}, \mathcal{B}|\mathcal{C})}{P(\mathcal{A})P(\mathcal{B})} \frac{P(\mathcal{A}', \mathcal{B}'|\mathcal{C}')}{P(\mathcal{A}')P(\mathcal{B}')} P(\mathcal{C}, \mathcal{C}'|E) \end{aligned} \tag{5.9}$$

It is shown in [2] that the first two terms are

$$\frac{P(\mathcal{A}, \mathcal{B}|\mathcal{C})}{P(\mathcal{A})P(\mathcal{B})} = (1-\alpha)^{2n} \frac{1}{(k_d k_l)^n}, \tag{5.10}$$

$$\frac{P(\mathcal{A}', \mathcal{B}'|\mathcal{C}')}{P(\mathcal{A}')P(\mathcal{B}')} = (1-\alpha)^{2n'} \frac{1}{(k_d k_l)^{n'}}. \tag{5.11}$$

Joint conditional probability $P(\mathcal{C}, \mathcal{C}'|E)$ can be seen as two vehicle trajectories stitch at rendezvous and form a combined trajectory. We know $P(C_1) = 1/n_v$ given there are n_v possible solutions, and $P(C_2|C_1) = 1/n_b$ because there are n_b neighbors of C_1. And the combined trajectory has n_c unique observations where

$$\max(n, n') \leq n_c \leq (n + n'). \tag{5.12}$$

By induction,

$$P(\mathcal{C}, \mathcal{C}'|E) = \frac{1}{n_b^{n_c - 1}} \frac{1}{n_v}. \tag{5.13}$$

Plugging (5.10)-(5.13) into (5.9), we obtain the lemma. □

$P(\mathcal{C}, \mathcal{C}'|\mathcal{A}, \mathcal{B}, \mathcal{A}', \mathcal{B}', E)$ reflects how fast the conditional probability increase as query

graph grows when combining inputs from two vehicles. Its counterpart in single vehicle localization is $P(C|\mathcal{A},\mathcal{B})$ from [2]. Comparing the two, we have the following theorem.

Theorem 1. *The C-GBPL algorithm converges to actual localization is at least as fast as GBPL, because*

$$P(\mathcal{C},\mathcal{C}'|\mathcal{A},\mathcal{B},\mathcal{A}',\mathcal{B}',E) \geq P(C|\mathcal{A},\mathcal{B}) \tag{5.14}$$

Proof. For single vehicle, it is shown in [2] that the conditional probability that $M = \{\Theta, D\}$ is the true matching sequence given $Q = \{\Theta_q, D_q\}$ matches M is,

$$P(C|\mathcal{A},\mathcal{B}) = [(1-\alpha)^2 k_d k_l]^n \frac{1}{n_b^{n-1}} \frac{1}{n_v}. \tag{5.15}$$

Comparing (5.7) with (5.15), we have

$$\frac{P(\mathcal{C},\mathcal{C}'|\mathcal{A},\mathcal{B},\mathcal{A}',\mathcal{B}',E)}{P(C|\mathcal{A},\mathcal{B})} = \left[(1-\alpha)^2(k_d k_l)\right]^{n'} n_b^{n_c-n} \geq 1, \tag{5.16}$$

Because $k_l > \frac{1}{1-\alpha}$ and $k_d > \frac{1}{1-\alpha}$ are generally true, $n_b > 1$, and $n_c - n \geq 0$ according to (5.12). □

The minimum value of n_c happens when vehicle trajectories are the same. The maximum value of n_c happens when vehicles have non-overlapping trajectories. When $n' = 0$, we have $n_c = n$ and thus (5.16) has a ratio of 1. In such case, there is no gain on localization efficiency. However, for most cases, $P(\mathcal{C},\mathcal{C}'|\mathcal{A},\mathcal{B},\mathcal{A}',\mathcal{B}',E)$ is much bigger than $P(C|\mathcal{A},\mathcal{B})$ which results in significant increase in localization speed.

5.4 Experiments

We have implemented the proposed C-GBPL method using MATLAB and validated the algorithm in both simulation and physical experiments. The experiments are based on a map of College Station, Texas, U.S. which is 3.24 km^2 in area with 52.7 km roads.

The map is from OSM and termed as CSMap. We pre-process it to the modified HLG with 1102 nodes. For comparison purpose, we evaluate its performance against GBPL [2], which is the single vehicle localization counterpart.

5.4.1 Simulation

5.4.1.1 Data Generation

We track the localization performance of the 25 seed vehicles as vehicles of interests and gradually increase other vehicles on the street from a total of 25 to 1000 vehicles on the map. It simulates sparse to moderately dense traffic. At 1000 vehicles in CSMap, it means that the mean car-space is around 53 meters. All simulation results are the statistics of the 25 seed vehicles.

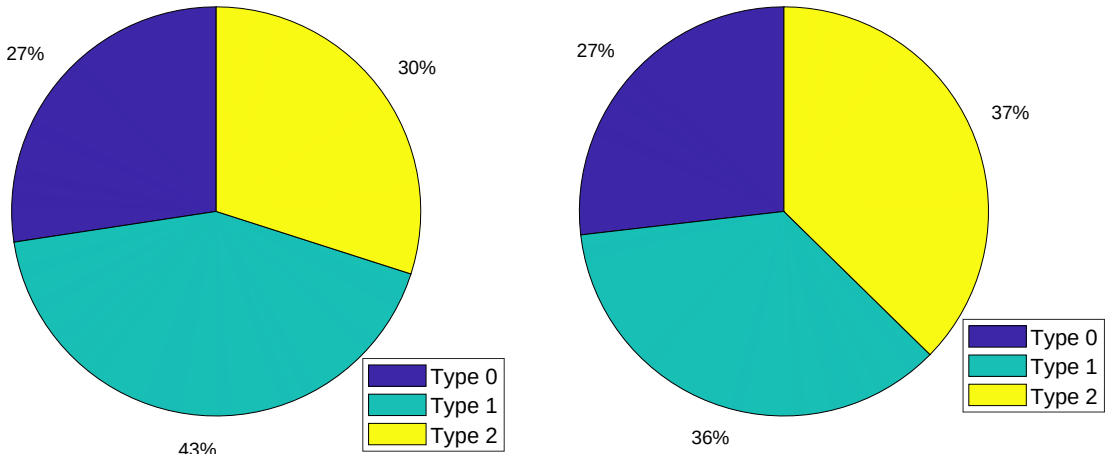


Figure 5.7: Pie chart of rendezvous events. (a) Trajectory type: random walk. (b) Trajectory type: same-street. Reprinted with permission from [3] © 2021 IEEE.

To generate vehicles trajectories, each vehicle starts at random vertices at the same time. Vehicles take two different driving strategies: *random-walk* and *same-street*. In the

random-walk strategy, the vehicle takes random turns at intersections. For the same-street strategy, the vehicle keeps driving on the same street. In reality, a vehicle's driving behavior is somewhere between the two extreme types. Fig. 5.7(a) and Fig. 5.7(b) show rendezvous event distribution for random walk and same-street for 1000 simulated vehicles, respectively. It is interesting that random walk generate more type 1 events (i.e. vehicles meet at the same street with different direction) while same street generates more type 2 events (vehicles meet at intersections).

5.4.1.2 Localization Comparison

To compare the two algorithms, we first compute the failure rate. A localization failure occurs when the algorithm fails to converge to a unique location. It may happen due vehicle trajectory or map itself. For example, if a city map consists of perfect square grid everywhere, our algorithm will fail. Fig. 5.8 shows the failure percentage for same-street strategy. We omit the failure percentage of random walk strategy because in this setting all vehicles can be localized. It is expected because random walk generates more unique query graphs. From Fig. 5.8, it is clear that C-GBPL utilizes the information from other vehicles and hence reduces failure rate to zero as traffic increases.

When GBPL and C-GBPL algorithms successfully localize the vehicle, it is also important to compare how fast the localization process takes. we use n which is the number of straight segments in the query to measure *localization speed*, because it tells how many inputs it takes to localize the vehicle. For a given n, the algorithm may provide multiple solutions if there are many similar routes in the map. If the number of solutions is one, then the vehicle is uniquely localized. Fig. 5.9 illustrate n's mean and $\pm\sigma$ range for both algorithms under the two driving strategies where σ is the standard deviation of n. Again, C-GBPL outperforms GBPL with a smaller n. The mean value n for C-GBPL decreases as traffic increases while the mean value for n for GBPL remains unchanged.

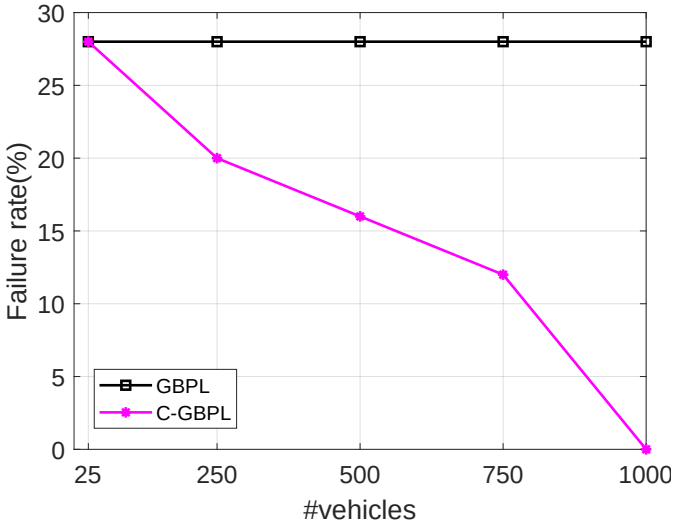
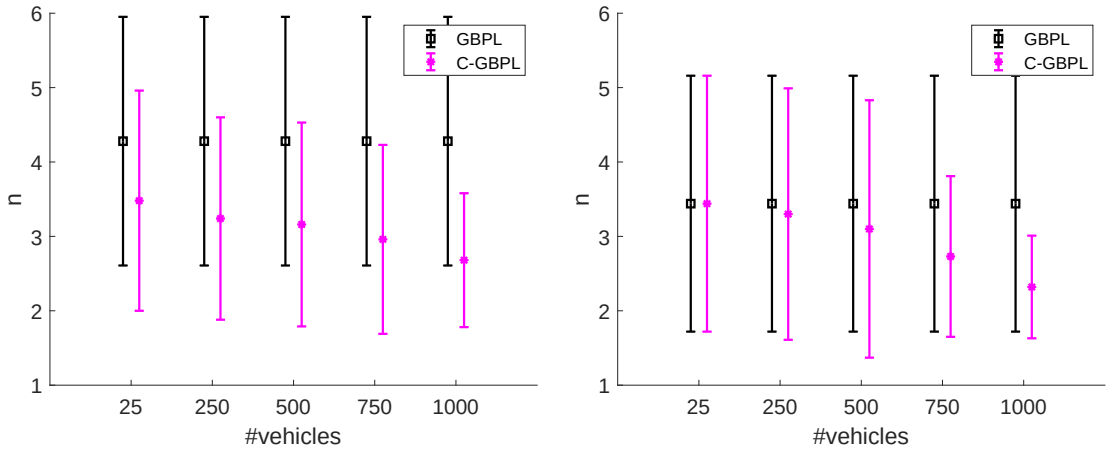


Figure 5.8: Failure rate percentage of trajectory type: same-street. Reprinted with permission from [3] © 2021 IEEE.



(a) (b)

Figure 5.9: Localization speed comparison. Each marker position is the mean, and each vertical segment is its corresponding $1 \pm \sigma$ range. (a) Trajectory type: random walk. (b) Trajectory type: same-street. Reprinted with permission from [3] © 2021 IEEE.

5.4.2 Physical Experiments

We compare GBPL and C-GBPL in physical experiment and test on different rendezvous scenarios. We collect three sequences which correspond to the trajectories of three vehicles. We record IMU readings at 400Hz and compass readings at 50Hz using a Google Pixel™ phone. Also, we access vehicle speed readings at 46.6Hz in average use an panda OBD-II Interface which provide velocity feedback from wheel encoder. To test C-GBPL, we vary different rendezvous time of these three vehicles and test on six rendezvous scenarios as shown in Fig. 5.10. We use #sols (number of solutions) for a given n to measure algorithm efficiency. For a given n, if the algorithm has fewer #sols than the other algorithm, the better in efficiency.

Tab. 5.1 shows the comparison between GBPL and C-GBPL in both localization efficiency and speed in terms of $\#sols$ and n accordingly. In summary, in all tests C-GBPL has better efficiency and with a speedup factor of 1.6x on average.

Table 5.1: LOCALIZATION SPEED AND EFFICIENCY. © 2021 IEEE.

			GBPL		C-GBPL	
Fig. 5.10 (Case)	Vehicle (Ego, the other)	CRSV (Type)	Speed (n)	Efficiency (#sols)	Speed (n)	Efficiency (#sols)
(a)	(car3, car1)	Type 2	5	122	**1**	**1**
(b)	(car2, car1)	Type 0	6	5	**5**	**1**
(c)	(car2, car1)	Type 2	6	3	**4**	**1**
(d)	(car1, car2)	Type 2	5	9	**3**	**1**
(e)	(car1, car2)	Type 0	5	9	**4**	**1**
(f)	(car2, car1)	Type 0	6	3	**4**	**1**

5.5 Conclusion

To assist vehicles in extreme weather or extreme environmental conditions, we developed the C-GBPL method that did not rely on the perception and recognition of external

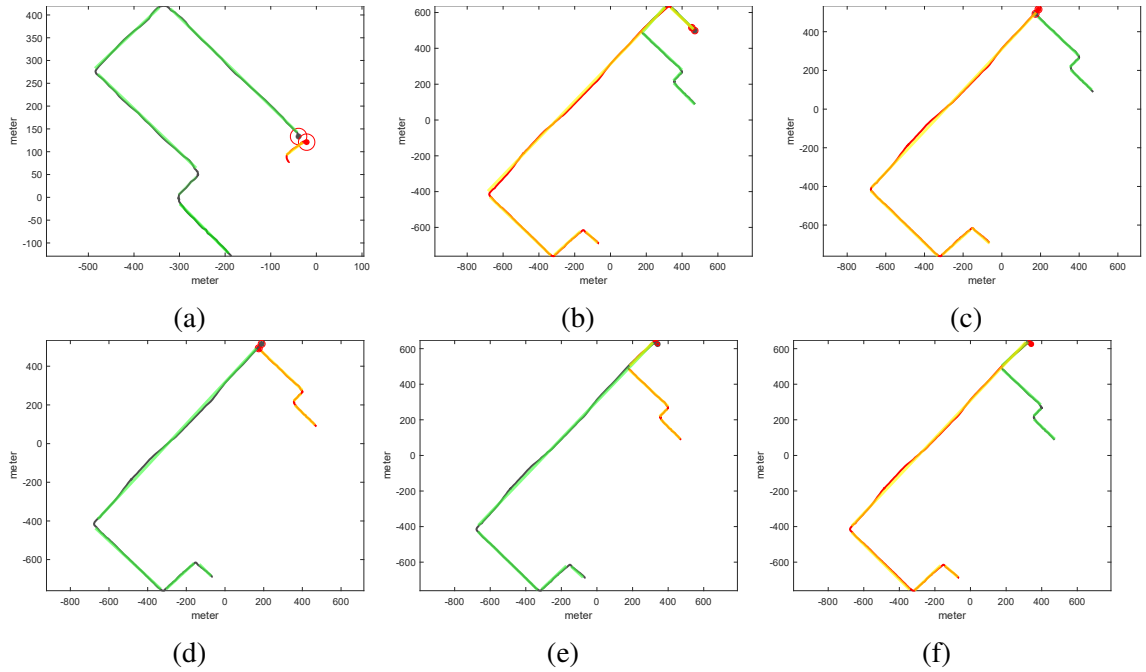


Figure 5.10: Rendezvous of three vehicles under different scenarios. We show the ego vehicle trajectory in red and mark straight segments in translucent yellow. The other vehicle trajectory is shown in gray and straight segments in translucent green. Reprinted with permission from [3] © 2021 IEEE.

landmarks to localize robots/vehicles in urban environments. The method was a multiple vehicle/robot collaborative localization scheme using V2V communication which combines features from rendezvous vehicles to accelerate the mapping process. We identified different rendezvous events to form the merged query graph. We performed graph-to-graph matching by aggregating vehicle prior beliefs and trim candidate vertex. We proved that the collaborative localization strategy is faster than its single vehicle counterpart in general cases. The algorithm was tested in both simulation and physical experiments and show superior performance over the single vehicle counterpart.

6. LOCALIZATION IN INCONSISTENT WIFI ENVIRONMENT*

In this Chapter, we propose a PL method in indoor environment with WiFi assistance. This is because indoor localization has become more important in recent years as mobile robots and mobile device users often need to find their locations where global positioning system (GPS) signals are unavailable. One low-cost solution is to utilize WiFi signals in the environment. The number of WiFi access points (APs) has dramatically increased in the past few years. A large number of APs are temporarily generated by cellphones and other mobile devices. Moreover, more and more infrastructural APs are equipped with beamforming capabilities which adjust radiation patterns according to client locations. These APs have large variation in their signal fields. We name those APs as inconsistent APs. Fig. 6.1 shows an example of a WiFi environment with inconsistent APs which dramatically change received signal strength (RSS) patterns. When the client cannot interrogate APs for their whereabouts and signal pattern changes, the existing WiFi localization approaches cannot handle inconsistent WiFi environments well. Their assumption of small variations in RSS spatial distribution is broken because a majority of APs may be inconsistent.

Building on existing WiFi fingerprinting approach, our method also employs Gaussian processes (GPs) to establish belief functions from priorly collected WiFi reference data. However, our approach utilizes two important designs to handle inconsistent APs. First, majority voting is introduced to the initial matching phase which allows us to develop a statistical hypothesis test to filter out inconsistent APs that are obvious out of places. Second, we use a windowed approach by employing a window of recent RSS readings along

*Reprinted with permission from Springer Nature Customer Service Centre GmbH: "Localization in Inconsistent WiFi Environment" by Hsin-Min Cheng and Dezhen Song 2016. *International Symposium on Robotics Research (ISRR)*, Copyright © 2017.

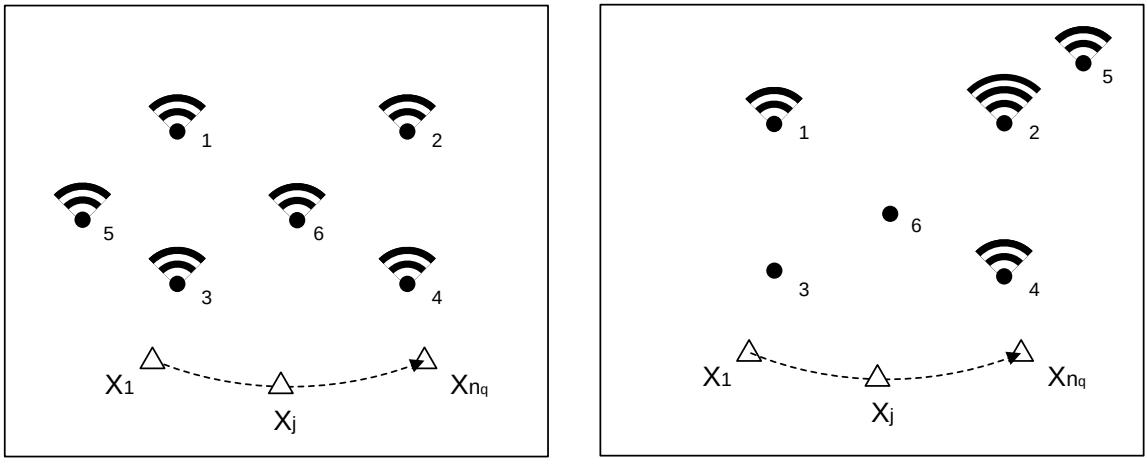


Figure 6.1: Example of localization scenarios. Left: APs and their RSS readings at reference collection time. Right: APs and their RSS readings at localization time. Even though the robot's trajectory is identical, the RSS pattern may be very different which leads to unsatisfying localization results. Reprinted with permission from [59].

with relative motion information provided by inertial measurement units (IMU) to develop posterior distribution of location. We formally derive the conditional distribution and determine the length of the time window by minimizing Shannon entropy. At last, we apply the maximum likelihood estimation (MLE) method to obtain refined localization results. We have implemented our algorithm and compared it with the state of the art k-Nearest-Neighbor (k-NN) approach. The experimental results show that our method outperforms its counterpart in inconsistent WiFi environments. Specifically, our algorithm achieves a mean localization error of less than 3.7 meters when 70% of APs are inconsistent.

6.1 Related Work

Our work is related to the simultaneous localization and mapping (SLAM) [12] using on-board sensors. SLAM using a lidar [24] and/or a camera [14, 23, 78–80] can be more accurate but is computation intensive and suffers from reliability issues and specific requirements for environments. WiFi localization has its own advantages when considering

sensor size, power, and cost. Recently, researchers have applied the graph-SLAM structure using WiFi and other wireless signals [81–83]. These methods provide good localization results, but the inconsistent WiFi environments have not been considered as SLAM in dynamic environments remains a difficult problem [84, 85]. SLAM approaches usually assume stationary environment/landmark locations. The underlying assumption that WiFi APs or RSS patterns can be treated as stationary landmarks is no longer true under inconsistent WiFi environments. In fact, inconsistent WiFi environments are highly dynamic instead of just containing a few moving landmarks in a largely stationary background.

WiFi localization has been a popular research area [90, 104, 113]. We can classify the existing methods into four types: angle based, time based, RSS modeling based, and fingerprinting based approaches (see Tab. 6.1). Angle based approaches use AP with multiple antennas to compute Angle of Arrival (AoA) of the multi-path signals received at each AP and localize through triangulation. Time based approaches include time of arrival (ToA) or time difference of arrival (TDoA) relies on the propagation time of signals traveling from a transmitter to a receiver. However, precise time synchronization is required which is not available using commodity WiFi. Both the two methods relies on known AP locations, and the client needs to interrogate APs for their locations or precise synchronization. Although more accurate in general, these two methods bear high cost in infrastructure and are difficult to be deployed. The RSS modeling based approaches model WiFi RSS signal

Table 6.1: OVERVIEW OF WIFI INDOOR LOCALIZATION METHODS. REPRINTED WITH PERMISSION FROM [59].

Method	Hardware cost	Known AP Locations	Communications b/w AP and client	System/Solution
Angle based (AoA)	High	Y	Y	[86–89]
Time based(ToA/ TDoA)	High	Y	Y	[90–94]
RSS modeling based	Low	Y	N	[74, 95–101]
Fingerprint based	Low	N	N	[83, 102–112]

propagation in the space assuming known AP locations, which have advantages of low cost and easy deployment. These methods often suffer from low accuracy because signal attenuation is often complex and hard to be predicted. Fingerprinting based approaches require priorly-mapped WiFi RSSs in the working space to construct a database for localization purpose which does not require communication between APs and the client or known AP locations. Our approach inherits the benefits with focus on addressing the dynamic signal pattern issues.

In order to improve WiFi localization accuracy, auxiliary sensors are combined into the above approaches, such as cameras, which are used to recognize landmarks, and IMUs for motion estimation [108, 114–116]. The sensor configuration in our approach is the same as the latter. Existing work in the IMU-assisted WiFi localization systems [117, 118] localize pedestrians by utilizing step count information to mitigate IMU drifting issue. However, the step count information is not available for robots and these methods have not explicitly consider inconsistent WiFi environments.

To deal with uncertainties in WiFi environment, existing works detect outliers using k-NN [119, 120], penalize RSS readings from uncertain APs, and signify APs with strong RSS readings in the k-NN method. Laoudias et al. [120] assume the fault model of AP with on and off status and set threshold on sum of distance of k-NN method to mitigate errors from faulty APs. These works have demonstrated the advantages of removing outliers. However, they use only RSS at the current moment which may be challenging for moving users who experience signal fluctuations. We take a windowed approach by using sequence matching which greatly improves the robustness to inconsistent APs. AP selection is not a new technique in WiFi localization. However, the main focus is to reduce the computation cost and improve accuracy instead of handling inconsistent WiFi environments. Existing approaches choose APs with the highest RSS observation [121] or select APs based on entropy-based information gain criterion [122]. They cannot be guaranteed the selected

APs are reliable in inconsistent WiFi environments due to different design purposes.

6.2 Problem Formulation

6.2.1 Application Scenario and Assumptions

Our system extends the aforementioned fingerprinting methods, which assume that there is a mobile robot equipped with SLAM ability to pre-scan the environment to establish a database of WiFi RSS readings and their corresponding listening locations. The database is referred to as the WiFi reference data thereafter. It is clear that the reference data contain inconsistent AP RSSs.

At client side, the robot also perceives WiFi signal strengths and accumulates them over time which are referred to as WiFi query data. It is true that WiFi query data may also contain noisy data from inconsistent APs. The focus here is the client side localization problem in the presence of inconsistent AP signals in both query and reference data. Each localization client is equipped with an IMU. The client may be a mobile robot or a cellphone user. To focus on the most relevant issues, we have the following assumptions:

1. WiFi and IMU readings have been synchronized and time-stamped.

2. The RSS reception noises are Gaussian and IMU signal noises are white.

It is also worthing noting that we do not assume that we start with a known initial position and hence our localization is a global location problem instead of an incremental localization problem.

6.2.2 Notations and Problem Definition

Common notations are defined as follows,

- Reference data: $\mathbf{D} = \{(\mathbf{z}_{r,i}, X_{r,i}) | i = 1, \cdots, n\}$ is composed of n input-output pairs where i is the index variable, $X_{r,i}$ is the position in $2D$ or $3D$ Cartesian coordinate

system, and vector $\mathbf{z}_{r,i}$ contains the WiFi RSS readings at $X_{r,i}$. Subscript r refers to the reference data. $\mathbf{z}_{r,i}$'s dimension is determined by number of APs in the environment. Let us define AP index $m \in M$ where $M := \{1, \cdots, m_{\max}\}$ is the AP index set. There are $m_{\max} = |M|$ distinct APs.

- Query data: subscript q refers to query data. Let j be the time index where $j \geq 0$. Vector $\mathbf{z}_{q,j}$ contains all WiFi RSS readings at time j. The sequence of WiFi query data from the beginning to time j is denoted as

$$\mathbf{z}_{q,0:j} = \{\mathbf{z}_{q,0}, \cdots, \mathbf{z}_{q,j}\}. \tag{6.1}$$

- IMU readings from the beginning to time j are denoted as $\mathbf{a}_{0:j}$ and $\boldsymbol{\omega}_{0:j}$ for accelerations and angular velocities, respectively. Note that $\mathbf{a}_{0:j}$ and $\boldsymbol{\omega}_{0:j}$ contain a lot more entries than that of query set in (6.1) due to high sampling frequency.

With important notations defined, let us formulate the localization problem,

Problem 4. *Given* $\mathbf{D}$*,* $\mathbf{z}_{q,0:j}$*,* $\mathbf{a}_{0:j}$*, and* $\boldsymbol{\omega}_{0:j}$ *at time* j*, estimate location* X_j *.*

6.3 System Design and Algorithm

Our system architecture is illustrated in Fig. 6.2 which contains four main blocks shaded in gray: 1) Reference pre-processing where we construct RSS-location belief model using the reference data, 2) Query pre-processing where we match a time window of query APs with reference data and reconstruct the corresponding window of relative motion of the client using IMU inputs, 3) Query AP selection where we match query WiFi with those in the reference data to remove unmatched query APs, and 4) Location estimation where we fuse the recent relative motion with historical RSS to localize the client

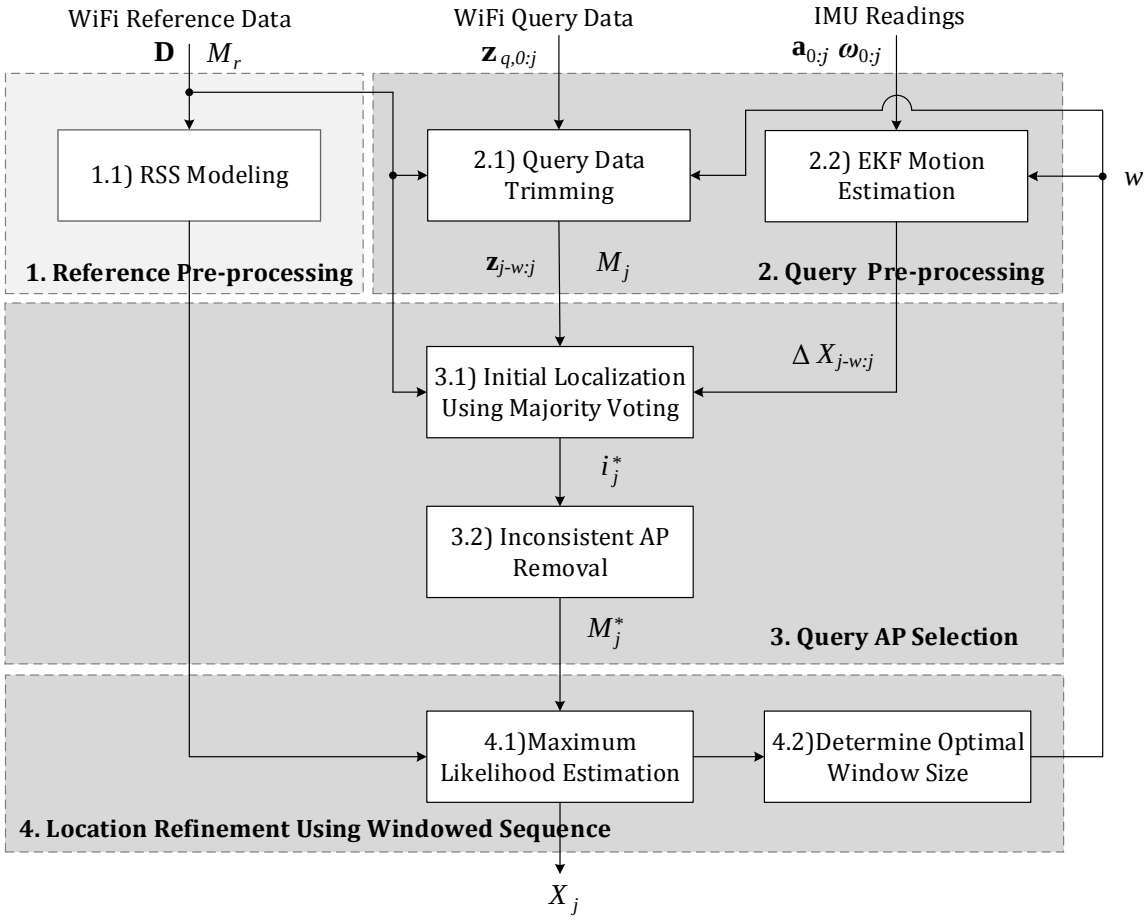


Figure 6.2: System Diagram: the light gray region needs only one time computation and all the dark gray regions are computed for each frame. Reprinted with permission from [59].

and determine the optimal window size for next time frame. We begin with reference pre-processing (Box 1 of Fig. 6.2) where we construct an RSS-location belief model from the WiFi reference data using GPs.

6.3.1 Reference Pre-processing

Let $X_{r,i}$ be i-th location where reference data is collected and $z_{r,i,m}$ be the perceived RSS for the m-th AP at this location. $X_{r,i}$ can be either 2D or 3D depending on the environment. Let $\mathbf{X}_r := [X_{r,1}, \cdots, X_{r,n}]^\mathsf{T}$ be a location matrix containing all collection

locations in the reference data $\mathbf{D}$. Define $\mathbf{z}_{r*,m} := [z_{r,1,m} \cdots, z_{r,n,m}]$ as all RSSs for the m-th AP in the reference data. Then the combined data set $\mathbf{D}_m := \{\mathbf{z}_{r*,m}, \mathbf{X}_r\}$ be the training set of the m-th AP to instantiate a GP to characterize a regression model $f_m(\cdot)$ between $\mathbf{X}_r$ and $\mathbf{z}_{r*,m}$. For each element in $\mathbf{z}_{r*,m}$,

$$z_{r,i,m} = f_m(X_{r,i}) + \epsilon, \tag{6.2}$$

where $\epsilon \sim \mathcal{N}(0, \sigma_{nm}^2)$ is the observation noise and σ_{nm}^2 is the variance. For two different locations $X_{r,i}$ and $X_{r,i'}$, we employ the kernel function $k_m(\cdot)$ in GP to characterize the correlation between their function values, namely, $f_m(X_{r,i})$ and $f_m(X_{r,i'})$:

$$k_m(X_{r,i}, X_{r,i'}) = \sigma_{fm}^2 \exp(-\frac{1}{2\lambda_m^2}|X_{r,i} - X_{r,i'}|^2), \tag{6.3}$$

where σ_{fm}^2 is the signal variance and λ_m is the length scale. The covariance matrix $\mathbf{K}_m$ is an $n \times n$ matrix with the (i, i')-th element $\mathbf{K}_m(i, i') = k_m(X_{r,i}, X_{r,i'})$ where $i, i' \in \{1, \cdots, n\}$. For the m-th AP, the predicted function value $\tilde{z}$ for an arbitrary location X_a conditioned on $\mathbf{X}_r$ and $\mathbf{z}_{r*,m}$ is

$$\tilde{z} = K_a^{\mathsf{T}}(\mathbf{K}_m + \sigma_{nm}^2 I)^{-1}\mathbf{z}_{r*,m}, \tag{6.4}$$

where *I* is an n-dimensional identity matrix and K_a is an $n \times 1$ vector which captures the relationship between X_a and $\mathbf{X}_r$ using (6.3), $K_a(i, 1) = k_m(X_{r,i}, X_a)$ where $i \in \{1, \cdots, n\}$. The values of parameters σ_{fm}^2, σ_{nm}^2 and λ_m are learned using hyperparameter estimation mentioned in [83]. Since GP is a zero mean process, we subtract the mean of $\mathbf{z}_{r*,m}$ before training and add it back after to get the required $\tilde{z}$.

We can do this for each AP and hence we can obtain a location belief model based on all APs in the reference data.

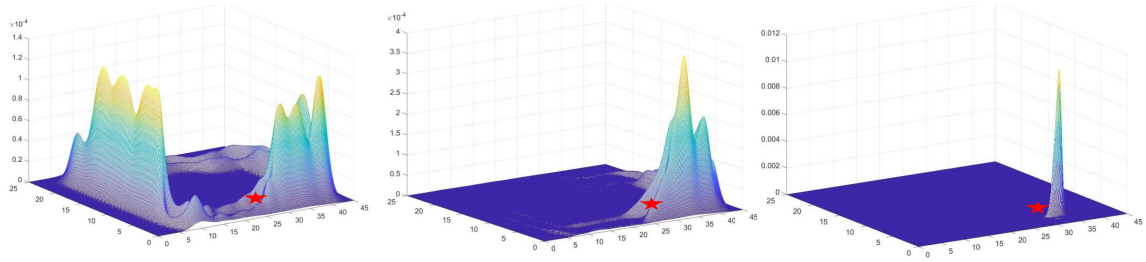
6.3.2 Query Pre-processing

Before we match query data with the reference data, we need to remove query APs that do not exist in the reference data. It is clear that their RSSs would not assist localization. Subsequently, we reorganize $\mathbf{z}_{q,j}$ in (6.1) to $\mathbf{z}_j$ by trimming out the useless RSSs. Define the surviving AP index set as M_j. Similarly, we also update the historic query data set $\mathbf{z}_{0:j} := \{\mathbf{z}_0, \cdots, \mathbf{z}_j\}$ from $\mathbf{z}_{q,0:j}$. In fact, it is not necessary to employ the entire historic query data set for localization computation. We only need to backtrack a window of length w readings, which allow us to establish the windowed query data set $\mathbf{z}_{j-w:j}$ from time $j-w$ to time j. We will discuss how to determine the optimal window length in Section 6.3.5.

Next, we associate the relative motion from time $j-w$ to j for the query data $\mathbf{z}_{j-w:j}$. Although we do not know the absolute position, we can utilize IMU data to obtain the relative motion in the time window. We define $\Delta X_{a:b} = X_b - X_a$ as the travel distance between time a and time b. To get $\Delta X_{a:b}$, we employ an EKF-based approach for IMUs [123–125]. From the EKF, $\Delta X_{a:b}$ follows Gaussian distribution with a mean of $\overline{\Delta X_{a:b}}$ and a covariance of $\Sigma_{\Delta X_{a:b}}$. Define the relative motion set $\Delta\mathbf{X}_{j-w:j} := \{\Delta X_{j-k+1:j-k} | k = 1, \cdots, w\}$, which captures the relative motion within the time window.

6.3.3 Query AP Selection

With $\mathbf{z}_{j-w:j}$ and $\Delta\mathbf{X}_{j-w:j}$ obtained, we can perform statistical testing to remove inconsistent APs which refers to APs that may have changed locations or have significant signal strength changes. If we do not remove inconsistent APs, localization results may suffer. In fact, we can visualize this issue. We can obtain posterior distribution of X_j directly using GPs in (6.4) based on $\mathbf{z}_j$ which is shown in Fig. 6.3(a). A desirable outcome is supposed to be a unimodal distribution with its peak close to or overlapping with the actual location. Unfortunately, the inconsistent APs have lead to a multimodal distribution and the ground truth position does not correspond to a peak position which indicated localization would



(a) (b) (c)

Figure 6.3: Sample cases of posterior condition probability of robot/client location. The color changing from yellow to blue corresponds to high to low probability regions, respectively. The red star is the ground truth location. The x-axis and y-axis that span horizontal plane are 2D Cartesian coordinates, and vertical z-axis represents probability. (a) Directly computed from GPs without removing inconsistent APs and using windowed inputs. (b) After inconsistent AP removal. (c) After inconsistent AP removal and location refinement. Reprinted with permission from [59].

fail if such belief model is naively used.

6.3.3.1 Initial Localization Using Majority Voting

To remove inconsistent APs, we perform an initial low resolution localization by simply assuming current location X_j is co-located at each location $X_{r,i}$ in reference set and then we can compare the sequence of RSSs. Relative motion set $\Delta\mathbf{X}_{j-w:j}$ allow us to generate assumed reference location information $\tilde{\mathbf{X}}_{r,i} = \{X_{r,i}\} \cup \{X_{r,i} - \sum_{p=1}^{k} \Delta X_{j-p:j-p+1} | k = 1, \cdots, w\}$ for the entire window. Plug $\tilde{\mathbf{X}}_{r,i}$ into (6.4), we obtain the assumed reference RSSs $\tilde{\mathbf{z}}_{r,i}$. Note that $\tilde{\mathbf{z}}_{r,i}$ is of the same length as $\mathbf{z}_{j-w:j}$.

Now we find the most possible location by matching query $\mathbf{z}_{j-w:j}$ to reference $\tilde{\mathbf{z}}_{r,i}$. We find best match location from initial location candidates using majority voting over best location candidates identified by each AP.

For the m-th AP at time $j-k$, its query entry in $\mathbf{z}_{j-w:j}$ is $z_{j-k,m}$ and the corresponding reference entry in $\tilde{\mathbf{z}}_{r,i}$ is $\tilde{z}_{i,m,k}$. The matching metric function f_E is the summation of the

squared l^2-norm over the window,

$$f_E(m,i) = \sum_{k=0}^{w} (\tilde{z}_{i,m,k} - z_{j-k,m})^2. \tag{6.5}$$

The m-th AP can predict best location i_m by comparing f_E values over the entire reference set

$$i_m = \arg\min_{i \in \{1,\cdots n\}} f_E(m,i).$$

Combining the best solutions for all APs, we have a candidate solution set $I_m := \{i_m | m = 1, \cdots m_{\max}\}$. We employ majority voting to find the most agreed location index i_j^* as the solution. Specifically, we define a binary ballot function b_b:

$$b_b(i_m, i) = \begin{cases} 1, & i = i_m, \\ 0, & \text{otherwise.} \end{cases}$$

The location with the most votes wins,

$$i_j^* = \arg\max_{i \in I_m} \sum_{m=1}^{m_{\max}} b_b(i_m, i). \tag{6.6}$$

Now we know that the actual location is close to the location at reference index i_j^*. This information can be exploited to filter out inconsistent APs.

6.3.3.2 Inconsistent AP Removal

We develop statistical hypothesis testing to remove inconsistent APs. To perform the statistics testing, we begin with analyzing $f_E(m, i_j^*)$ as a distribution. We would like to derive the probability that $f_E(m, i_j^*)$ is abnormally large. For the m-th AP, have two

hypotheses:

H_0 : The m-th AP is an inconsistent AP vs. H_1 : The m-th AP is a consistent AP

The significance level is α. We reject H_0 if $p-$value is less than α.

Let $\tilde{z}^*_{m,k}$ be the reference entry for the m-th AP corresponds to location i^*_j. From the GP model, we know $(\tilde{z}^*_{m,k} - z_{j-k,m})/\sigma_{nm} \sim \mathcal{N}(0,1)$ is a random variable following the normal distribution with zero mean and a variance of 1. We know that $f_E(m, i^*_j)/\sigma^2_{nm}$ is the sum of squares of multiple normal distributions according to (6.5), $f_E(m, i^*_j)/\sigma^2_{nm}$ has to follow χ^2-distribution with $w+1$ degrees of freedom. Its cumulative probability function is $F(x, w+1) = \dfrac{\gamma((w+1)/2, x/2)}{\Gamma((w+1)/2)}$ where $\gamma(a,x) = \int_0^x t^{a-1}e^{-t}dt$ and $\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}dt$. The probability of a value from χ^2-distribution distribution is larger than x is

$$P\{f_E(m, i^*_j)/\sigma^2_{nm} \geq x\} = 1 - F(x, w+1) = \alpha. \tag{6.7}$$

Setting $F(x, w+1) = 1-\alpha$ allows us to find threshold $x = F^{-1}(1-\alpha)$ where $F^{-1}(\cdot)$ is the quantile function defined as $F^{-1}(a) = inf\{y : F(y) \geq a\}$. Thus we reject H_0 if $f_E(m, i^*_j) \leq \sigma^2_{nm}F^{-1}(1-\alpha)$. After the statistical testing, we remove inconsistent APs. The remaining AP index set is defined as M^*_j. We trim $\mathbf{z}_{j-w:j}$ accordingly. Fig. 6.3(b) illustrates how inconsistent AP removal help reshape the posterior location distribution which has its highest peak closer to the actual AP location. However, inconsistent AP removal cannot distinguish APs which change their positions slightly. Therefore, we still have a multi-modal distribution with many peaks, which limits the localization accuracy. To handle this issue, we introduce localization refinement.

6.3.4 *Location Refinement Using Windowed Sequence*

Our idea is to derive the posterior probability of X_j for present time j given RSS readings $\mathbf{z}_{j-w:j}$ and relative motion information $\Delta\mathbf{X}_{j-w:j}$ and then apply MLE method to obtain the location estimation (Fig. 6.2 Box 4).

Let $P(X_j|\mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j})$ be the posterior probability of X_j given $\mathbf{z}_{j-w:j}$ and $\Delta\mathbf{X}_{j-w:j}$. With $\Delta\mathbf{X}_{j-w:j}$, the robot/client position X_{j-k} can be obtained (see Fig. 6.4),

$$X_{j-k} = X_j - \Delta X_{j-k} = X_j - \sum_{p=1}^{k} \Delta X_{j-p+1:j-p}. \tag{6.8}$$

Assuming $X_j = x_j$, then the conditional distribution of X_{j-k} given (x_j, ΔX_{j-k}) is,

$$\mathtt{X}_{j-k}|x_j, \Delta X_{j-k} = x_j - \Delta X_{j-k}, \tag{6.9}$$

where operator '$\cdot|\cdot$' represents the condition for a random variable with conditions at the right side of '|'. Since ΔX_{j-k} is obtained from EKF based on IMU inputs, it follows Gaussian distribution with a mean of $\overline{\Delta X_{j-k}}$, and a covariance matrix of $\Sigma_{\Delta X_{j-k}}$:

$$X_{j-k}|x_j, \Delta X_{j-k} \sim \mathcal{N}(x_j - \overline{\Delta X_{j-k}}, \Sigma_{\Delta X_{j-k}}). \tag{6.10}$$

Now let us derive the posterior probability using Bayes theorem,

$$P(X_j|\mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j}) = \frac{P(X_j, \mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j})}{P(\mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j})}. \tag{6.11}$$

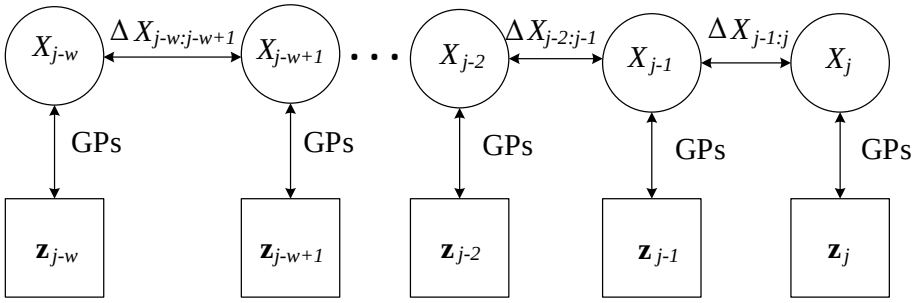


Figure 6.4: Relationships between locations, displacements, and RSS readings in the window of prior locations and RSSs. Each prior location can be associated to its RSS readings using GPs. Reprinted with permission from [59].

We decompose $P(X_j, \mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j})$ by taking $\mathbf{z}_j$ to the front,

$$\begin{aligned} & P(X_j, \mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j}) \\ &= P(\mathbf{z}_j | X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-1}) P(X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-1}) \\ &= P(\mathbf{z}_j | X_j) P(X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-1}). \end{aligned} \tag{6.12}$$

This is because the current observation $\mathbf{z}_j$ only depends on the current location. Again, we decompose $P(X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-1})$

$$\begin{aligned} & P(X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-1}) \\ &= P(\mathbf{z}_{j-1} | X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-2}) P(X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-2}) \\ &= P(\mathbf{z}_{j-1} | X_{j-1} = X_j - \Delta X_{j,j-1}) P(X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-2}). \end{aligned} \tag{6.13}$$

This is because $\mathbf{z}_{j-1}$ is independent of $\mathbf{z}_{j-w:j-2}$ given X_{j-1}.

For $P(X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-2})$, we have

$$\begin{aligned} & P(X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-2}) \\ &= P(\mathbf{z}_{j-2}|X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-3})P(X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-3}) \\ &= P(\mathbf{z}_{j-2}|X_{j-2} = X_j - \sum_{p=1}^{2} \Delta X_{j-p+1,j-p})P(X_j, \Delta\mathbf{X}_{j-w:j}, \mathbf{z}_{j-w:j-3}). \end{aligned} \tag{6.14}$$

X_{j-2} and X_j are related using (6.8). By decomposing $P(X_j|\mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j})$ iteratively, we rewrite (6.11) as

$$P(X_j|\mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j}) = \frac{P(\mathbf{z}_j|X_j)\prod_{k=1}^{w} P(\mathbf{z}_{j-k}|X_{j-k} = X_j - \sum_{p=1}^{k} \Delta X_{j-p+1,j-p})}{P(\mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j})}. \tag{6.15}$$

We integrate displacement $\Delta\mathbf{X}_{j-k:j}$ for each term:

$$P(\mathbf{z}_{j-k}|X_{j-k} = X_j - \sum_{p=1}^{k} \Delta X_{j-p+1,j-p}) = \int P(\mathbf{z}_{j-k}|x_{j-k})f(\Delta x_{j-k})d(\Delta x_{j-k}), \tag{6.16}$$

where $f(\Delta x_{j-k})$ is a Gaussian distribution function (6.10) obtained from EKF result.

Under the GP model, the probability distribution for the m-th AP's RSS conditioned on $X_j = x_j$ is

$$z_{j,m}|X_j \sim \mathcal{N}(\mu_m(x_j, \mathbf{D}), \Sigma_m(x_j, \mathbf{D})), \tag{6.17}$$

where $\mu_m(x_j, \mathbf{D}) = k_{X_j}^T(K_m + \sigma_{nm}^2 I)^{-1}\mathbf{z}_{r,*m}^T$ and $\Sigma_m(x_j, \mathbf{D}) = k(x_j, x_j) - k_{X_j}^T(K_m + \sigma_{nm}^2 I)^{-1}k_{X_j}$ with $k(x_j, x_j) = \sigma_{fm}^2$ obtained from (6.3). Since RSSs of all APs are independent, we have

$$f(\mathbf{z}_j|X_j = x_j) = \prod_{m \in M_j^*} f(z_{j,m}|X_j = x_j). \tag{6.18}$$

Similarly, we can obtain a probability distribution function for $\mathbf{z}_{j-k}|X_{j-k}$. With results from (6.16) and (6.18) and the fact that $\mathbf{z}_{j-w:j}$ and $\Delta\mathbf{X}_{j-w:j}$ are independent, we can compute (6.15) to obtain $P(X_j|\mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j})$. Fig. 6.3(c) illustrates the posterior distribution for the windowed inputs which appears as a unimodal distribution with its peak located close to the actual location. This significantly increases the accuracy of the localization algorithm.

Finally, we estimate X_j by applying MLE to the posterior probability in (6.15),

$$\hat{X}_j = \arg\max_{X_j} P(\mathbf{z}_j|X_j) \int \prod_{k=1}^{w} P(\mathbf{z}_{j-k}|X_{j-k} = X_j - \sum_{p=1}^{k} \Delta X_{j-p+1,j-p}) d(\Delta x_{j-k}). \tag{6.19}$$

Note that we dropped the denominator in (6.15) because it is not a function of X_j.

6.3.5 Determine Optimal Window Size

The remaining issue is how to determine the optimal window size w of RSS sequence. It is worth noting that increasing window size w significantly increases computation load. Also, the relative motion information $\Delta\mathbf{X}_{j-w:j}$ has its variance increasing over time due to IMU drifting and eventually becomes useless due to its large spatial uncertainty. To choose an appropriate window size, we minimize the Shannon entropy over window size. Define A as a latticed set of the localization space. The lattice resolution is 0.1 meters in each dimension in our settings. Define $H(w, j)$ as the Shannon entropy over the probability distribution $P(X_j|\mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j})$,

$$H(w, j) = -\sum_{X_j \in A} P(X_j|\mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j}) \log P(X_j|\mathbf{z}_{j-w:j}, \Delta\mathbf{X}_{j-w:j}).$$

We choose the optimal w that minimizes $H(w, j)$ over all possible window length set $\mathbf{w} := \{0, 1, ..., w_{\max}\}$ where $w_{\max}$ is the maximum allowable window size that covers the

entire A. Then we find the optimal solution w^*,

$$w^* = \arg\min_{w \in \mathbf{w}} H(w, j). \tag{6.20}$$

The resulting w^* will be used as the window size for time frame $j + 1$.

6.4 Experiments

We have implemented our algorithm using MATLAB under a PC with Windows 7. We evaluate our method using real world data from physical experiments. Three algorithms are compared in our experiment.

- MLE-S: Our complete method including both AP removal and MLE refinement,
- MLE-NS: our algorithm without AP removal in Section 6.3.3.2. We test this variant of our method to show the benefit of inconsistent AP removal, and
- k-NN [115]: This method is chosen as the counterpart because it has ability to handle some degrees of inconsistent WiFi environments despite it only targets at handling RSS fluctuations. It also employs IMU to assist WiFi localization. For brevity, we name it as k-NN method here.

Dataset: We have collected datasets from three different buildings: H. R. Bright building (HRBB), Scoates Hall (SCTS), and Wisenbaker Engineering Building (WEB) at Texas A&M University using a Nexus 7 tablet and a mobile robot (see Fig. 6.5). Our robot is equipped with a Hokuyo UTM-30LX lidar and provides 2D lidar map (see Fig. 6.5 (b)) with timestamps. The location from lidar-based SLAM is used as a ground truth with an error of less than 10 centimeters. The dataset consists of WiFi RSS readings collected at 1Hz and IMU readings at 100Hz. Tab. 6.2 describes details of each dataset including the overall trajectory, the number of APs in reference data, and query data. We have collected

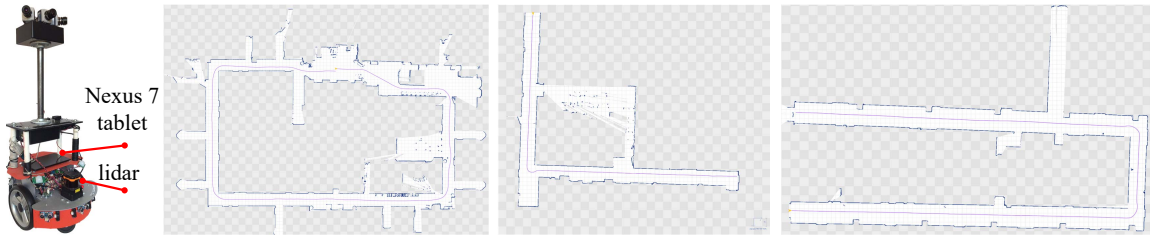


Figure 6.5: (a) Our data collection robot and sensor configuration. 2D lidar maps are used as ground truth: (b) HRBB, (c) SCTS, and (d) WEB. Reprinted with permission from [59].

the reference data and query data in different days. Therefore, the two data sets do not share the exact same number of APs. The shared AP number are shown in the last row of the Tab. 6.2.

Table 6.2: DATASET DESCRIPTION. REPRINTED WITH PERMISSION FROM [59].

Building	HRBB	SCTS	WEB
Trajectory (meters)	96	50	80
Reference Data # of AP	252	132	159
Query Data AP # of AP	246	135	171
Shared APs between two data sets	195	130	158

Evaluation Metric: The localization error at time j is defined as the Euclidean distance between the estimation $\hat{X}_j$ and ground truth $X_{g,j}$: $\varepsilon_j = ||\hat{X}_j - X_{g,j}||$. The error is measured in meters.

Tab. 6.3 shown the test results. The best results are highlighted in bold fonts. We have tested all three algorithms under different percentage of inconsistent APs ranging from 0% to 90% in the environment. To generate inconsistent APs, we first use systematic random sampling on the shared APs between reference data and query data and shift AP RSS

Table 6.3: MEAN LOCALIZATION ERROR (METERS). REPRINTED WITH PERMISSION FROM [59].

		Inconsistent APs									
Dataset	Method	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
HRBB	k-NN	3.3	3.5	3.8	4.2	4.4	5.5	6.5	8.6	9.7	15.1
	MLE-NS	**3.2**	3.4	3.9	4.2	4.3	5.6	7.0	6.7	7.3	10.0
	MLE-S	**3.2**	**3.2**	**3.2**	**3.2**	**3.2**	**3.2**	**3.2**	**3.5**	**4.6**	**9.1**
SCTC	k-NN	**1.6**	1.8	1.9	2.6	3.0	3.8	5.8	7.1	8.0	10.4
	MLE-NS	2.1	2.1	2.1	2.6	2.9	3.6	6.9	6.6	6.4	**6.4**
	MLE-S	1.7	**1.6**	**1.7**	**1.7**	**2.1**	**2.0**	**2.4**	**3.1**	**4.9**	10.1
WSB	k-NN	**1.2**	**1.3**	**1.3**	1.5	2.1	2.9	6.2	8.9	12.7	16.3
	MLE-NS	1.6	1.5	1.6	1.7	1.9	2.3	4.4	6.4	**8.9**	**14.5**
	MLE-S	1.4	1.5	1.4	**1.4**	**1.4**	**1.6**	**2.3**	**3.7**	10.3	19.4

patterns randomly to form inconsistent APs. It is clear that our method is more robust than the counterpart under inconsistent WiFi environments. Also, the inconsistent AP removal process does help in maintain consistent localization accuracy up to 70% inconsistent APs. Our methods achieve the best or close to the best mean localization error as long as the inconsistent AP ratio is less than 70%. At 80% or 90% inconsistent AP ratio, our proposed method does not work well, this is because signal to noise ratio is too low and inconsistent AP removal process may fail.

Fig. 6.6 shows the effectiveness of inconsistent AP removal scheme by illustrating true positive (TP) rates, which is defined as

$$TP = \frac{\text{\#consistent AP selected}}{\text{\#All selected APs}},$$

over different inconsistent AP ratios. Our method ensures that TP rates remain above 0.5 under 70% inconsistent APs environment among different datasets with a mean localization error less than 3.7 meters.

The last evaluation is runtime of our approach. From We report the average runtime for each time frame are 0.30 seconds on average for SCTS dataset, 0.89 seconds for HRBB

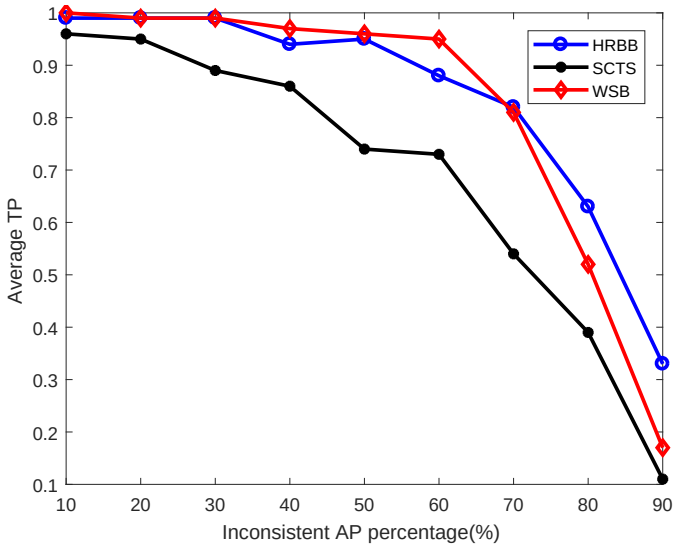


Figure 6.6: Average TP rates of selected APs after the AP removal process. Reprinted with permission from [59].

dataset, and 0.63 seconds for WEB dataset. From alogithm analysis, the most time consuming operation is computing location posterior probability which is proportional to the number of APs (Tab. 6.2.). Our runtime result confirm it. Recall that the time interval for each time frame is 1 second in experiment set up, the runtime for localization is tolerable.

6.5 Conclusion

We reported a WiFi-based localization method designed to handle inconsistent WiFi environments where mobile APs and APs with beamforming capabilities cause significant changes in radiation patterns in RSSs. Building on the WiFi fingerpinting method that utilizes GPs to establish a belief model from reference data, our method employed majority voting along with embedded statistical hypothesis testing to remove inconsistent APs. Instead of using RSS pattern at a single time instance, we used a historical sequence to further improve the robustness to inconsistent APs. We derived a posterior location distribution function for the sequence and applied MLE to refine localization results. Our

method was tested and compared to an existing method. The results showed that our method outperforms the counterpart and our design is effective. In the future, we will conduct more experiments and comparison studies. We will also focus on analysis and improving computation speed.

7. CONCLUSION AND FUTURE WORK

In this study, we investigated PL approaches as fallback solutions to enhance localization robustness under conditions that may challenge all exteroceptive sensors. We explore PL approaches in two application scenarios: urban areas and indoor environments.

7.1 Conclusion: PL approaches in urban areas

We proposed three PL approaches in urban area application. In Chapter 3, we proposed a minimalist intermittent heading-based approach which employed a gyroscope and a compass along with a prior map. The PLAM method extracted long and straight road segment headings and pre-processed it into a heading graph. The information from the sensors was used to obtain heading changes during traveling. To localize the robot, the heading sequence was matched to the heading graph in a Bayesian framework that tracks both sensor and map uncertainties. We introduced entropy to investigate map properties and studied how the PLAM method performs under maps with different entropies.

In Chapter 4, we developed the GBPL method to estimate a rudimentary vehicle trajectory computed from an IMU, a compass, and a wheel encoder and matched it with a prior map. This method improves localization robustness in a degenerated map and enables continuous localization. To address the drifting issue in the dead-reckoning process and the fact that the vehicle trajectory may not overlap with road waypoints on the map, we developed a heading-length feature sequence graph-based matching approach. Once the map matching is successful, our algorithm tracks vehicle movement and uses the map information to regulate EKF's drifting issue.

In Chapter 5, we developed a collaborative localization scheme to remove the dependency of vehicle trajectory and improve localization efficiency. We developed the C-GBPL method which was a multiple vehicle/robot collaborative localization scheme using V2V

communication. The C-GBPL method combines features from rendezvous vehicles to accelerate the mapping process. We identified different rendezvous events to form the merged query graph. We performed graph-to-graph matching by aggregating vehicle prior beliefs and trim candidate vertices. We proved that the collaborative localization strategy is faster than its single vehicle counterpart in general cases. The algorithm was tested in both simulation and physical experiments and showed superior performance over the single vehicle counterpart.

To conclude, we designed graph-based matching algorithms using propioceptive sensors in urban areas to enhance localization robustness. The superiority of our PL approaches was demonstrated on urban datasets by comparing with state-of-the-art methods.

7.2 Conclusion: PL approaches in indoor environments

In Chapter 6, we explored a PL approach in an indoor environment with WiFi to handle inconsistent WiFi environments with mobile APs and APs with beamforming capabilities. Building on the WiFi fingerprinting method that utilizes GPs to establish a belief model from reference data, our method employed majority voting along with embedded hypothesis testing to remove inconsistent APs. Instead of using RSS patterns at a single time instance, we used the relative motion information provided by IMU to further improve the robustness to inconsistent APs. We derived a posterior location distribution function for the sequence and applied MLE to refine localization results. The results showed that our method outperforms the counterpart and our design is effective to localize the robot in inconsistent wifi environment.

7.3 Future Works

Our work is the beginning toward PL with a prior map. We discuss the future work from the following perspectives.

- **Embedding into exteroceptive methods**. The PL method is served as a fall-back

solution when exteroceptive sensors cannot work properly. To ensure a seamless switch between GPS or other exteroceptive localization methods to PL is a very interesting future topic. As an example, since both our method and existing GPS/ inertial navigation system (INS) rely on the INS system, they can be naturally integrated. Also, PL methods can be combined with visual place recognition to propose a human-like localization for robots. As mentioned by early researchers [126], robots do not always need precise location information. In the future, localization methods that can work with local metrical and global topological maps can be considered.

- **Other trajectory features**. The proposed method uses straight and long segments as features in graph-matching in Chapter 3, 4, and 5. However, when the trajectory contains only continuous curves, the proposed methods cannot deal with it. In this case, it is interesting to consider designing curve segments as features. On the other hand, in this study we mainly use single straight and long segments as features. Other high-level representation of trajectory features can also be explored. For example, such as finding distinguishable road sequences in the map and pre-processed the map with additional layers. Adding trajectory features can assist the localization method in robustness and increase the localization speed.

- **Planning for localization**. In the proposed method, it is assumed that the robot is able to navigate in the environment and make turns at appropriate locations to assist localization. As one direction of future study, the path planning for localization can be explored so the robot can localize itself more efficiently. For example, a systematic way of generating turns may be employed. A possible direction is using the maximum probability framework to find best turns. Also, for some task-oriented robots the path planning and localization tasks need to be carefully considered si-

multaneously. A planning strategy that the robot can find its location while not sacrificing the task performance is also an interesting topic.

REFERENCES

[1] H. Cheng, D. Song, A. Angert, B. Li, and J. Yi, "Proprioceptive localization assisted by magnetoreception: A minimalist intermittent heading based approach," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 586–593, 2019.

[2] H. Cheng and D. Song, "Graph-based proprioceptive localization using a discrete heading-length feature sequence matching approach," *IEEE Transactions on Robotics*, 2020.

[3] H. M. Cheng, C. Chou, and D. Song, "Vehicle-to-vehicle collaborative graph-based proprioceptive localization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 990–997, 2021.

[4] H. Aly, A. Basalamah, and M. Youssef, "Accurate and energy-efficient GPS-less outdoor localization," *ACM Trans. Spatial Algorithms Syst.*, vol. 3, pp. 4:1–4:31, July 2017.

[5] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[6] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, pp. 1–22, 2019.

[7] S. Chitta, E. G. Jones, M. Ciocarlie, and K. Hsiao, "Perception, planning, and execution for mobile manipulation in unstructured environments," *IEEE Robotics and Automation Magazine, Special Issue on Mobile Manipulation*, vol. 19, no. 2, pp. 58–71, 2012.

[8] J. Borenstein and Y. Koren, "Obstacle avoidance with ultrasonic sensors," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 213–218, 1988.

[9] D. An and H. Wang, "VPH: A new laser radar based obstacle avoidance method for intelligent mobile robots," in *Fifth World Congress on Intelligent Control and Automation*, vol. 5, pp. 4681–4685, 2004.

[10] Z. Chen and X. Huang, "End-to-end learning for lane keeping of self-driving cars," in *IEEE Intelligent Vehicles Symposium (IV)*, pp. 1856–1860, 2017.

[11] M. W. Spong and M. Vidyasagar, *Robot dynamics and control*. John Wiley & Sons, 2008.

[12] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[13] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[14] Y. Lu and D. Song, "Visual navigation using heterogeneous landmarks and unsupervised geometric constraints," *IEEE Transactions on Robotics (T-RO)*, vol. 31, pp. 736–749, June 2015.

[15] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.," in *Robotics: Science and Systems*, vol. 2, 2014.

[16] J. Civera, O. G. Grasa, A. J. Davison, and J. Montiel, "1-point RANSAC for EKF-based structure from motion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3498–3504, 2009.

[17] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real time localization and 3D reconstruction," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 363–370, 2006.

[18] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM," *Robotics: Science and Systems VI*, vol. 2, no. 3, p. 7, 2010.

[19] M. A. Brubaker, A. Geiger, and R. Urtasun, "Map-based probabilistic visual self-localization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 652–665, April 2016.

[20] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 1–19, 2016.

[21] G. Floros, B. van der Zander, and B. Leibe, "OpenStreetSLAM: Global vehicle localization using OpenStreetMaps," in *IEEE International Conference on Robotics and Automation*, pp. 1054–1059, May 2013.

[22] N. Radwan, G. D. Tipaldi, L. Spinello, and W. Burgard, "Do you see the bakery? leveraging geo-referenced texts for global localization in public maps," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4837–4842, May 2016.

[23] N. Carlevaris-Bianco and R. M. Eustice, "Learning visual feature descriptors for dynamic lighting conditions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2769–2776, 2014.

[24] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, pp. 206–211, 2003.

[25] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D lidar SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*,

pp. 1271–1278, 2016.

[26] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4372–4378, 2010.

[27] T. Hunter, P. Abbeel, and A. Bayen, "The path inference filter: model-based low-latency map matching of probe vehicle data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 507–529, 2014.

[28] C. Fouque and P. Bonnifait, "Matching raw GPS measurements on a navigable map without computing a global position," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 887–898, 2012.

[29] Y. Cui and S. S. Ge, "Autonomous vehicle positioning with GPS in urban canyon environments," *IEEE transactions on robotics and automation*, vol. 19, no. 1, pp. 15–25, 2003.

[30] I. Constandache, R. Roy, and C. I. Rhee, "CompAcc: Using mobile phone compasses and accelerometers for localization," in *In INFOCOM*, 2010.

[31] C. Chou, A. Kingery, D. Wang, H. Li, and D. Song, "Encoder-camera-ground penetrating radar tri-sensor mapping for surface and subsurface transportation infrastructure inspection," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1452–1457, May 2018.

[32] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[33] J. Yi, J. Zhang, D. Song, and S. Jayasuriya, "IMU-based localization and slip estimation for skid-steered mobile robots," in *IEEE International Conference on Intel-*

ligent Robots and Systems, pp. 2845–2850, 2007.

[34] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV navigation and localization: A review," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131–149, 2013.

[35] W. Kang and Y. Han, "SmartPDR: Smartphone-based pedestrian dead reckoning for indoor localization," *IEEE Sensors journal*, vol. 15, no. 5, pp. 2906–2916, 2014.

[36] G. C. Karras, S. G. Loizou, and K. J. Kyriakopoulos, "On-line state and parameter estimation of an under-actuated underwater vehicle using a modified dual unscented kalman filter," in *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pp. 4868–4873, 2010.

[37] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 425–437, 2002.

[38] L. Huang, B. He, and T. Zhang, "An autonomous navigation algorithm for underwater vehicles based on inertial measurement units and sonar," in *International Asia Conference on Informatics in Control, Automation and Robotics*, vol. 1, pp. 311–314, 2010.

[39] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[40] P. Merriaux, Y. Dupuis, P. Vasseur, and X. Savatier, "Fast and robust vehicle positioning on graph-based representation of drivable maps," in *IEEE International Conference onRobotics and Automation (ICRA)*, pp. 2787–2793, 2015.

[41] P. Ruchti, B. Steder, M. Ruhnke, and W. Burgard, "Localization on OpenStreetMap data using a 3D laser scanner," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5260–5265, 2015.

[42] R. Jiang, S. Yang, S. S. Ge, H. Wang, and T. H. Lee, "Geometric map-assisted localization for mobile robots based on uniform-Gaussian distribution," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 789–795, 2017.

[43] Y. Jin and Z. Xiang, "Robust localization via turning point filtering with road map," in *IEEE Intelligent Vehicles Symposium (IV)*, pp. 992–997, 2016.

[44] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ." `https://www.openstreetmap.org`, 2017.

[45] Google Maps contributors. `https://www.google.com/maps/`, 2017.

[46] M. Quddus and S. Washington, "Shortest path and vehicle trajectory aided map-matching for low frequency GPS data," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 328–339, 2015.

[47] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[48] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 2564–2571, 2011.

[49] D. Tedaldi, A. Pretto, and E. Menegatti, "A robust and easy to implement method for IMU calibration without external equipments," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3042–3049, May 2014.

[50] M. Kok and T. B. Schön, "Magnetometer calibration using inertial sensors," *IEEE Sensors Journal*, vol. 16, pp. 5679–5689, July 2016.

[51] R. Spica and P. R. Giordano, "Active decentralized scale estimation for bearing-based localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5084–5091, 2016.

[52] V. Pierlot and M. Van Droogenbroeck, "A new three object triangulation algorithm for mobile robot positioning," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 566–577, 2014.

[53] J. M. Font and J. A. Batlle, "Mobile robot localization. revisiting the triangulation methods," *IFAC Proceedings Volumes*, vol. 39, no. 15, pp. 340–345, 2006.

[54] D. Xu, H. Badino, and D. Huber, "Topometric localization on a road network," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3448–3455, 2014.

[55] S. Funke, R. Schirrmeister, S. Skilevic, and S. Storandt, "Compass-based navigation in street networks," in *International Symposium on Web and Wireless Geographical Information Systems*, pp. 71–88, Springer, 2015.

[56] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[57] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[58] J. Yi, H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu, "Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation," *IEEE Transactions on Robotics*, vol. 25, pp. 1087–1097, Oct 2009.

[59] H.-M. Cheng and D. Song, "Localization in inconsistent WiFi environments," in *The International Symposium on Robotics Research (ISRR), Puerto Varas, Chile*, 2017.

[60] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proceedings IEEE International Conference on Data Mining*,

pp. 289–296, 2001.

[61] H. Shatkay and S. B. Zdonik, "Approximate queries and representations for large data sequences," in *Proceedings of the Twelfth International Conference on Data Engineering*, pp. 536–545, IEEE, 1996.

[62] A. Gudmundsson and N. Mohajeri, "Entropy and order in urban street networks," *Scientific reports*, vol. 3, p. 3324, 2013.

[63] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[64] G. Boeing, "Urban spatial order: Street network orientation, configuration, and entropy," *Applied Network Science*, vol. 4, no. 1, pp. 1–19, 2019.

[65] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361, 2012.

[66] J. Wahlström, I. Skog, J. G. P. Rodrigues, P. Händel, and A. Aguiar, "Map-aided dead-reckoning using only measurements of speed," *IEEE Transactions on Intelligent Vehicles*, vol. 1, pp. 244–253, Sep. 2016.

[67] B. Yu, L. Dong, D. Xue, H. Zhu, X. Geng, R. Huang, and J. Wang, "A hybrid dead reckoning error correction scheme based on extended Kalman filter and map matching for vehicle self-localization," *Journal of Intelligent Transportation Systems*, vol. 23, no. 1, pp. 84–98, 2019.

[68] R. A. Kronmal and A. V. Peterson Jr, "On the alias method for generating random variables from a discrete distribution," *The American Statistician*, vol. 33, no. 4, pp. 214–218, 1979.

[69] N. Mohajeri and A. Gudmundsson, "The evolution and complexity of urban street networks," *Geographical Analysis*, vol. 46, no. 4, pp. 345–367, 2014.

[70] J. Borenstein and L. Feng, "Correction of systematic odometry errors in mobile robots," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 569–574, 1995.

[71] S. I. Roumeliotis and G. A. Bekey, "Distributed multirobot localization," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 781–795, Oct 2002.

[72] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous robots*, vol. 8, no. 3, pp. 325–344, 2000.

[73] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[74] C. Kim, D. Song, Y. Xu, J. Yi, and X. Wu, "Cooperative search of multiple unknown transient radio sources using multiple paired mobile robots," *IEEE Transactions on Robotics*, vol. 30, pp. 1161–1173, Oct. 2014.

[75] K. Y. Leung, T. D. Barfoot, and H. H. Liu, "Decentralized localization of sparsely-communicating robot networks: A centralized-equivalent approach," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 62–77, 2009.

[76] L. Luft, T. Schubert, S. I. Roumeliotis, and W. Burgard, "Recursive decentralized collaborative localization for sparsely communicating robots," in *Proceedings of Robotics: Science and Systems*, June 2016.

[77] E. D. Nerurkar and S. I. Roumeliotis, "Asynchronous multi-centralized cooperative localization," in *IEEE/RSJ International Conference on Intelligent Robots and*

Systems, pp. 4352–4359, 2010.

[78] J. Civera, O. G. Grasa, A. J. Davison, and J. Montiel, "1-point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry," *Journal of Field Robotics*, vol. 27, no. 5, pp. 609–631, 2010.

[79] A. Majdik, D. Gálvez-López, G. Lazea, and J. A. Castellanos, "Adaptive appearance based loop-closing in heterogeneous environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1256–1263, 2011.

[80] Y. Lu and D. Song, "Robust RGB-D odometry using point and line features," in *IEEE International Conference on Computer Vision (ICCV), Santiago, Chile*, Dec. 2015.

[81] J. Huang, D. Millman, M. Quigley, D. Stavens, S. Thrun, and A. Aggarwal, "Efficient, generalized indoor WiFi GraphSLAM," in *IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*, pp. 1038–1043, May 2011.

[82] P. Mirowski, T. K. Ho, S. Yi, and M. MacDonald, "SignalSLAM: Simultaneous localization and mapping with mixed WiFi, bluetooth, LTE and magnetic signals," in *International Conference on Indoor Positioning and Indoor Navigation*, pp. 1–10, Oct 2013.

[83] B. Ferris, D. Fox, and N. Lawrence, "WiFi-SLAM using Gaussian process latent variable models," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence, Hyderabad, India*, Jan. 2007.

[84] D. Zou and P. Tan, "CoSLAM: Collaborative visual SLAM in dynamic environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 354–366, Feb 2013.

[85] D. H. Kim and J. H. Kim, "Effective background model-based RGB-D dense visual odometry in a dynamic environment," *IEEE Transactions on Robotics*, vol. 32, pp. 1565–1573, Dec 2016.

[86] S. Sen, J. Lee, K.-H. Kim, and P. Congdon, "Avoiding multipath to revive inbuilding WiFi localization," in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, ACM, 2013.

[87] J. Xiong and K. Jamieson, "ArrayTrack: A fine-grained indoor location system," in *(NSDI)*, pp. 71–84, USENIX, 2013.

[88] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, "SpotFi: Decimeter level localization using WiFi," *SIGCOMM Comput. Commun. Rev.*, vol. 45, pp. 269–282, Aug 2015.

[89] J. Gjengset, J. Xiong, G. McPhillips, and K. Jamieson, "Phaser: Enabling phased array signal processing on commodity WiFi access points," in *Proceedings of the 20th annual international conference on Mobile computing and networking*, pp. 153–164, ACM, 2014.

[90] I. Guvenc and C.-C. Chong, "A survey on TOA based wireless localization and NLOS mitigation techniques," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 3, 2009.

[91] K. R. Joshi, S. S. Hong, and S. Katti, "Pinpoint: Localizing interfering radios," in *NSDI*, USENIX, 2013.

[92] X. Li and K. Pahlavan, "Super-resolution TOA estimation with diversity for indoor geolocation," *IEEE Transactions on Wireless Communications*, vol. 3, no. 1, pp. 224–234, 2004.

[93] A. T. Mariakakis, S. Sen, J. Lee, and K.-H. Kim, "Sail: Single access point-based indoor localization," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pp. 315–328, ACM, 2014.

[94] D. Vasisht, S. Kumar, and D. Katabi, "Decimeter-level localization with a single WiFi access point.," in *NSDI*, 2016.

[95] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings of the IEEE international Conference on Computer Communications(INFOCOM), Tel-Aviv, Israel*, Mar. 2000.

[96] O. Serrano, J. M. Canas, V. Matellán, and L. Rodero, "Robot localization using WiFi signal without intensity map," in *International Workshop on Algorithmic Foundationsof Robotics (WAFR), Utrecht/Zeist, The Netherlands*, July 2004.

[97] K. Whitehouse, C. Karlof, and D. Culler, "A practical evaluation of radio signal strength for ranging-based localization," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, pp. 41–52, January 2007.

[98] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2010.

[99] H. Lim, L.-C. Kung, J. C. Hou, and H. Luo, "Zero-configuration indoor localization over IEEE 802.11 wireless infrastructure," *Wirel. Netw.*, vol. 16, FEB. 2010.

[100] D. Song, C. Kim, and J. Yi, "Simultaneous localization of multiple unknown CSMA-based wireless sensor network nodes using a mobile robot with a directional antenna," *Journal of Intelligent Service Robots*, vol. 2, pp. 219–233, Oct. 2009.

[101] D. Song, C. Kim, and J. Yi, "Simultaneous localization of multiple unknown and transient radio sources using a mobile robot," *IEEE Transactions on Robotics*,

vol. 28, pp. 668–680, June 2012.

[102] M. Ocana, L. Bergasa, M. Sotelo, J. Nuevo, and R. Flores, "Indoor robot localization system using WiFi signal measure and minimizing calibration effort," in *Proceedings of the IEEE International Symposium on Industrial Electronics, Dubrovnik, Croatia*, June 2005.

[103] S. Saha, K. Chaudhuri, D. Sanghi, and P. Bhagwat, "Location determination of a mobile device using IEEE 802.11b access point signals," in *IEEE Wireless Communications and Networking Conference(WCNC), New Orleans, LA*, March 2003.

[104] S. He and S. H. G. Chan, "Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 466–490, 2016.

[105] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, 2012.

[106] R. Nandakumar, K. K. Chintalapudi, and V. N. Padmanabhan, "Centaur: Locating devices in an office environment," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, 2012.

[107] C. Wu, Z. Yang, and Y. Liu, "Smartphones based crowdsourcing for indoor localization," *IEEE Transactions on Mobile Computing*, vol. 14, pp. 444–457, Feb 2015.

[108] J. Biswas and M. Veloso, "WiFi localization and navigation for autonomous indoor mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA), Anchorage, USA*, May 2010.

[109] A. Howard, S. Siddiqi, and G. S. Sukhatme, "An experimental study of localization using wireless ethernet," in *Field and Service Robotics*, pp. 145–153, Springer,

2003.

[110] A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, and D. S. Wallach, "Robotics-based location sensing using wireless ethernet," *Wireless Networks*, vol. 11, no. 1-2, pp. 189–204, 2005.

[111] B. Balaguer, G. Erinc, and S. Carpin, "Combining classification and regression for wifi localization of heterogeneous robot teams in unknown environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[112] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye, "Push the limit of WiFi based localization for smartphones," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2012.

[113] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1067–1080, 2007.

[114] W. W. L. Li, R. A. Iltis, and M. Z. Win, "A smartphone localization algorithm using RSSI and inertial sensor measurement fusion," in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2013.

[115] M. Jin, B. Koo, S. Lee, C. Park, M. J. Lee, and S. Kim, "IMU-assisted nearest neighbor selection for real-time WiFi fingerprinting positioning," in *International Conference onIndoor Positioning and Indoor Navigation (IPIN), Zurich, Switzerland*, Oct 2014.

[116] M. Quigley, D. Stavens, A. Coates, and S. Thrun, "Sub-meter indoor localization in unmodified environments with inexpensive sensors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2010.

[117] Z.-A. Deng, Y. Hu, J. Yu, and Z. Na, "Extended Kalman filter for real time indoor localization by fusing WiFi and smartphone inertial sensors," *Micromachines*, vol. 6, no. 4, pp. 523–543, 2015.

[118] Y. Zhuang and N. El-Sheimy, "Tightly-coupled integration of WiFi and MEMS sensors on handheld devices for indoor pedestrian navigation," *IEEE Sensors Journal*, vol. 16, no. 1, pp. 224–234, 2016.

[119] J. So, J.-Y. Lee, C.-H. Yoon, and H. Park, "An improved location estimation method for WiFi fingerprint-based indoor localization," *International Journal of Software Engineering and Its Applications*, vol. 7, no. 3, pp. 77–86, 2013.

[120] C. Laoudias, M. P. Michaelides, and C. G. Panayiotou, "Fault detection and mitigation in WLAN RSS fingerprint-based positioning," *Journal of Location Based Services*, vol. 6, no. 2, pp. 101–116, 2012.

[121] M. A. Youssef, A. Agrawala, and A. U. Shankar, "WLAN location determination via clustering and probability distributions," in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, March 2003.

[122] Y. Chen, Q. Yang, J. Yin, and X. Chai, "Power-efficient access-point selection for indoor location estimation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 877–888, July 2006.

[123] S. Sukkarieh, P. Gibbens, B. Grocholsky, K. Willis, and H. F. Durrant-Whyte, "A low-cost, redundant inertial measurement unit for unmanned air vehicles," *The International Journal of Robotics Research*, vol. 19, pp. 1089–1103, June 2000.

[124] G. Dissanayake, S. Sukkarieh, E. Nebot, and H. Durrant-Whyte, "The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for

land vehicle applications," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 731–747, Oct 2001.

[125] J. Yi, H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu, "Modeling and analysis of skid-steered mobile robots with applications to low-cost inertial measurement unit-based motion estimation," *IEEE Transactions on Robotics*, vol. 25, pp. 1087–1097, Oct. 2009.

[126] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli, "Local metrical and global topological maps in the hybrid spatial semantic hierarchy," in *IEEE International Conference on Robotics and Automation*, pp. 4845–4851, 2004.