Graph-based Proprioceptive Localization Using a Discrete Heading-Length Feature Sequence Matching Approach

Hsin-Min Cheng and Dezhen Song

Abstract—Proprioceptive localization 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 a camera or a laser ranger finder. These methods depend on proprioceptive sensors such as inertial measurement units (IMUs) and/or wheel encoders. Assisted by magnetoreception, the sensors can provide a rudimentary estimation of vehicle trajectory which is used to query a prior known map to obtain location. Named as graph-based proprioceptive localization (GBPL), we provide a low cost fallback solution for localization under challenging environmental conditions. 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 (HLG) abstracted from the prior known map to localize the robot under a graph-matching approach. Using the information from HLG, our location alignment and verification module compensates for trajectory drift, wheel slip, or tire inflation level. We have implemented our algorithm and tested it in both simulated and physical experiments. The algorithm runs successfully in finding robot location continuously and achieves localization accurate at the level that the prior map allows (less than 10m).

I. INTRODUCTION

Localization is a critical navigation function for vehicles or robots in urban area. Common localization methods employ global position system (GPS), a laser ranger finder, and a camera which are exteroceptive sensors relying on the perception and recognition of landmarks in the environment. However, high-rise buildings may block GPS signals. Poor weather and lighting conditions may challenge all exteroceptive sensors. What is needed is a fallback solution that enables vehicles to localize themselves under challenging conditions. This complements existing exteroceptive sensor-based localization methods. Inspired by biological systems, we combine proprioceptive sensors, such as inertial measurement units (IMU) 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. 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 (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).

The rest of the paper is organized as follows. After a review of related work in Section II, we define the problem in Section III. We introduce overall system design and detail GBPL in Section IV. We validate our system and algorithm with simulation and physical experiments in Section V and conclude the paper in Section VI.

II. 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 pro-
propiroceptive sensors. Exteroceptive sensors mainly rely on the perception and recognition of landmarks in the environment to estimate location. Mainstream exteroceptive sensors include cameras [2]–[4] and laser range finders [5]–[7]. These methods are often challenged by poor lighting conditions or weather conditions. GPS receiver [8], [9] 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 [10] and wheel encoders [11], 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 [12]. However, the sensor fusion approaches still strongly depend on exteroceptive sensor and cannot handle the 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 [13]. There are many applications using the dead reckoning approach such as autonomous underwater vehicles (AUVs) [14] and pedestrian step measurement [15], [16]. To estimate the state of the robot/vehicle, filtering-based schemes such as unscented Kalman filter (UKF) [17] and particle filter (PF) [18], [19] 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 [20] and modeling the wheel slip for skid-steered mobile robots [13]. 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 [2], [21]–[24]. According to [25], 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) [26] and Google Maps [27]. 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 [28]. The feature-based map is consisted with features of interest with its location. An example is ORB features [29] 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 [21], [30], [31], which focus on map-aided localization using proprioceptive sensors for mobile robots. In [30], 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 [21], 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 [31], 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.

### III. Problem Formulation

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) [26]. We have the following assumptions:

1. If needed, the vehicle is willing to change its course by making additional turns to assist our algorithm to find its location.
2. The robot is a nonholonomic system. i.e. it only performs longitudinal motion without lateral or vertical motions.
3. 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. Common notations are defined as follows:

- \( \mathcal{M}_p := \{x_m = [x_m, y_m]^T \in \mathbb{R}^2 | m \in \mathcal{M} \} \) represents the prior road map which is a set of GPS positions where \( \mathcal{M} \) 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.
- \( \mathbf{a} = \{a_j \in \mathbb{R}^3 | j = 0, 1, \ldots, N_j \} \) and \( \omega = \{\omega_j \in \mathbb{R}^3 | j = 0, 1, \ldots, N_j \} \) denote accelerometer readings and gyroscope angular velocities from the IMU, respectively.
- \( \phi = \{\phi_{j\phi} \in \mathbb{R} | j = 0, \ldots, \lfloor \frac{N_j}{c_{\phi}} \rfloor \} \) denotes compass readings where \( c_{\phi} \geq 1 \) since a compass often has lower sampling frequency than that of the IMU.
Problem 1. Given $M_p$, $a$, $\omega$, $\phi$ and $v$, localize the robot after its heading changes. As its localized, report robot location continuously.

IV. GBPL MODELING AND DESIGN

Our system diagram is illustrated in Fig. 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 $a$, $\omega$, $\phi$ and $v$ 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.

A. HLG Construction

We pre-process map $M_p$ to construct an HLG to facilitate heading-length matching. There are three reasons for using HLG instead of matching on $M_p$ directly.

- First, the vehicle trajectory may not exactly match with $M_p$. Since $M_p$ and most maps do not have lane-level information, the discrepancy between the estimated trajectory and $M_p$ is non-negligible which makes the direct trajectory-to-map matching unreliable. Fig. 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 $M_p$ directly is computationally expensive because the searching space grows with the total number of GPS waypoint positions in $M_p$.
- Third, the inevitably accumulated trajectory drift deteriorates the matching quality and makes the matching unreliable.

Fig. 2. System Diagram

Fig. 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.

Fig. 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.

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 in HG to construct HLG for heading-length matching. Fig. 4 illustrates an example. For completeness, we provide an overview here and more detail description of constructing the graph can be found in [1]. We denote the HLG by a directed graph \( \mathcal{M}_h = \{ \mathcal{V}_h, \mathcal{E}_h \} \) where \( \mathcal{V}_h \) is the vertex set and \( \mathcal{E}_h \) 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 \( e_{i,t} \in \mathcal{E}_h \) captures the connectivity between nodes and characterizes the orientation change between the two connected vertices \( v_i \) and \( v_t \). \( \mathcal{M}_h \) 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 vertices and short transitional segment vertices. The short length for vertices corresponding to those long straight segments; and two types of vertices: long straight segment and length vertices corresponding to those long straight road segments. Each vertex contains the following information:

\[
v_i = \{ x_i, \theta_i, d_i, b_i \},
\]

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

\[
d_i = \| x_{i,s} - x_{i,e} \|, \quad (2)
\]

and \( b_i = 1 \) is the binary variable indicate if the vertex is a long road segment. We only perform orientation estimation if \( d_i > t_i \) where \( t_i \) is the threshold for road segment length. That is,

\[
b_i = \begin{cases} 1, & d_i > t_i; \\ 0, & \text{otherwise}. \end{cases} \quad (3)
\]

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

The errors of GPS waypoints in each entry of \( X \) affect the accuracy of \( \theta_i \) and \( d_i \). To track map uncertainties caused by GPS errors, we derive the distribution of \( \theta_i \) and \( d_i \) using error variance propagation analysis [32]. 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 \( \sigma_g^2 \). According to [32], typical consumer grade navigation systems offer positional accuracy around \( \sigma_g = 10m \). The distribution of \( \theta_i \) that characterizes its uncertainty is

\[
\theta_i \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2), \quad (4)
\]

where \( \sigma_\theta^2 \) is derived in [1]. And the distribution of \( d_i \) is

\[
d_i \sim \mathcal{N}(\mu_d, \sigma_d^2) = \mathcal{N}(\mu_d, 2\sigma_g^2). \quad (5)
\]

B. 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.

1) EKF-based Trajectory Estimation: Note that readings from the IMU, the digital compass, and the vehicle velocity: \( a, \omega, \phi, \) and \( \nu \), are the inputs to the EKF-based approach to estimate vehicle trajectory [33]–[35]. To start the EKF, we need a stabilized initial compass reading \( \phi_0 \) to determine the initial vehicle orientation which can be obtained by driving on a long and straight segment of road (Assumption a.0). 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 \( X_{s,j} \) at time \( j \) be:

\[
X_{s,j} := [p_j, v_j, \Theta_j, s_j]^T, \quad (6)
\]

which includes position \( p_j = [x_j, y_j]^T \in \mathbb{R}^3 \), velocity \( v_j = [x, y, z] \in \mathbb{R}^3 \), and the Euler angles \( \Theta := [\alpha, \beta, \gamma]^T \) 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:

\[
\begin{align*}
p_j &= p_{j-1} + \tau_0 v_j \equiv p_j, \\
v_j &= v_{j-1} + \omega_0 (\dot{\mathbf{R}}(a) - \mathbf{G}) \\
\Theta_j &= \Theta_{j-1} + \tau_0 \dot{\mathbf{R}}(\omega) + c_{\gamma} \\
s_j &= s_{j-1},
\end{align*}
\]

where \( \tau_0 \) is the IMU sampling interval, \( \mathbf{G} = [0 \ 0 \ -9.8]^T \) is the gravitational vector, \( c_{\gamma} = [0 \ 0 \ \phi_0]^T \) is the initial orientation determined by \( \phi_0 \), \( \dot{\mathbf{R}} \) is the rotation matrix from \( \{ B \} \) to \( \{ I \} \), and \( \dot{\mathbf{E}} \) is the rotation rate matrix from \( \{ B \} \) to \( \{ I \} \).

For EKF observation models, we use velocity constraint from vehicle movement, sensory readings \( \phi \) and \( \nu \), and estimated scale by matching trajectory with map which will be discussed in Section IV-D3. First, according to Assumptions 1 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:

\[
(\ddot{\mathbf{R}})_{2,3} v_j = [0 \ 0]^T,
\]

where \( (\ddot{\mathbf{R}})_{2,3} \) is the second and third rows of \( \ddot{\mathbf{R}} \).
From the coordinate definition, the heading direction is \( \gamma \) defined in \( \{l\} \) (last component of \( \Theta^t \)), we take compass reading \( \phi \) 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 a threshold. With the cross-validated compass reading, we update heading direction \( \gamma \) by

\[
\gamma_j = \begin{cases} 
\phi_j, & \text{if } j = c_\phi j_
\phi, & \text{otherwise.}
\end{cases}
\] (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 [IV-D3]. For \( s \), we have

\[
s_j = s_{ssf},
\] (10)

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

\[
||v^j|| = \begin{cases} 
v^j, & \text{if } j = c_v j_v 
v^j_{j_v - 1}, & \text{otherwise.}
\end{cases}
\] (11)

Combining (9), (8), (11), and (10), we complete the observation model functions. The rest is to follow the standard EKF setup. Fig. 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.

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 are \( \gamma \) in \( \{l\} \) and is denoted by \( \gamma_{0,f} \). To obtain the query sequence, we segment \( \gamma_{0,f} \) to get stable headings and remove false positive headings that do not correspond to long and straight road segments. In Fig. 5(b), red horizontal segments are detected stable headings. Hence we obtain the set of query heading sequence which is denoted by \( \Theta_q = \{ \Theta_{q,k} | k = 1, \ldots, n \} \) 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 \( D_q = \{ d_{q,k} | k = 1, \ldots, n \} \) where \( d_{q,k} \) is the travel length of the segmented route (e.g. colored segments in Fig. 5(c)).

We denote the query heading-length sequence by \( Q := \{ \Theta_q, D_q \} \) which consists of the segmented heading-length sequence. The uncertainty of query sequence \( Q \) is obtained from EKF variance estimation. For \( \Theta_q \), we define \( \bar{\Theta}_{q,k} \) as the sample mean orientation of segment \( \Theta_{q,k} \) which contains \( n_{q,k} \) observations of random variable \( \theta_{q,k} \). \( \theta_{q,k} \) has a covariance matrix obtained from EKF. For \( D_q \), the variance of can also be derived from EKF and we denote variance of \( d_{q,k} \) by \( \sigma_{d_{q,k}}^2 \). 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 the localized computation. 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.

C. Global Localization Thread

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 \( \mathcal{V}_{h,f} \), 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.

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 \( Q := \{ \Theta_q, D_q \} = \{(\Theta_{q,k}, d_{q,k}) | k = 1, \ldots, n \} \), let us denote a candidate heading-length vertex sequence in \( \mathcal{M}_h \) by \( M := \{ \Theta, D \} = \{(\Theta, d) | k = 1, \ldots, n \} \) correspondingly. As a convention in this paper, for random vector \( * \), \( \mu_r \) represents its mean vector. Following the convention, mean matrix of \( Q \)
is defined as \( \mu_Q = [\mu_T^T, \mu_D^T]^T \) where \( \mu_T = [\mu_{T_1}, \ldots, \mu_{T_n}]^T \) and \( \mu_D = [\mu_{D_1}, \ldots, \mu_{D_n}]^T \). The mean matrix of \( M \) is denoted by \( \mu_M = [\mu_T^T, \mu_D^T]^T \) where \( \mu_T = [\mu_{T_1}, \ldots, \mu_{T_n}]^T \) and \( \mu_D = [\mu_{D_1}, \ldots, \mu_{D_n}]^T \).

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

\[
P(\mu_Q | \mu_M) = P(\mu_{T_1} | \Theta_{q_1}) \cdots P(\mu_{T_n} | \Theta_{q_n}) P(\mu_D | \Theta_{d_1}) P(\mu_D | \Theta_{d_2}),
\]

(12)

From [1], the conditional heading matching probability between \( \Theta_{q_1} \) and \( \Theta_{d_1} \) is

\[
P(\mu_{T_1} | \Theta_{q_1}) \propto \prod_{k=1}^n f_T(t(\theta_{q_1}, \theta_{k})),
\]

(13)

due to independent sensor noises and \( f_T(t(\theta_{q_1}, \theta_{k})) \) is the probability density function (PDF) of Student’s t-distribution. For length matching, the conditional matching probability between \( D_q \) and \( D \) is

\[
P(\mu_D | \Theta_{d_1}) = \prod_{k=1}^n f(z(d_{q,k}, d_k)),
\]

(14)

where \( f(\cdot) \) is the PDF of standard normal distribution, and \( z(d_{q,k}, d_k) = \sqrt{\frac{d_{q,k} - d_k}{\sigma_{q,k}^2 + \sigma_{d,k}^2}} \). Combining (13) and (14) and recalling that \( n \) is the number of straight segments in the query sequence, we rewrite (12) as follows,

\[
P(\mu_Q | \mu_M) \propto \prod_{k=1}^n f_T(t(\theta_{q_k}, \theta_k)) f(z(d_{q,k}, d_k)),
\]

(15)

3) Candidate Vertex Selection: To select on candidate vertices during matching, we perform statistical hypothesis testing to remove unlikely matchings. According to (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

\[
\begin{align*}
&H_0: [\mu_{q_k}, \mu_{d_k}]^T = [\mu_{\theta_k}, \mu_{d_k}]^T \\
&H_1: \text{otherwise.}
\end{align*}
\]

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 \( \alpha \) is a small probability. Both \( H_{0,\theta} \) and \( H_{0,d} \) are two-tailed distributions. We choose \( t_{\alpha/2,v} \) as the t-statistic with a cumulative probability of \( 1 - \frac{\alpha}{2} \) where \( v \) 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

\[
|t(\theta_{q_k}, \theta_k)| > t_{\alpha/2,v} \vee |z(d_{q_k}, d_k)| > z_{\alpha/2}.
\]

(17)

By sequentially applying the hypothesis testing on each corresponding pair \( (\theta_{q_k}, d_k), (\theta_{q_k}, d_{q_k}) \) from query sequence \( Q \) and candidate sequence \( M \) on HLG \( \mathcal{M}_h \), we determine whether \( M \) represents the actual trajectory. Fig. 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 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 (12) into two groups using the Ostu method [36]. 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) GL Algorithm: We summarize the heading-length matching method in Algorithm 1. 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 \( k = 1, \ldots, n \) is the length of the query sequence. The candidate vertex set is denoted by \( \mathcal{C}_k = \{v_i, \mathcal{V}_{M,\alpha, p_i} | i = 1, \ldots, n \mathcal{C}_k \} \), where each element in \( \mathcal{C}_k \) record the candidate vertex \( v_i \) (the starting vertex of the trajectory/path), \( \mathcal{V}_{M,\alpha} \) is the set of vertex path, and the matching probability \( p_i \) in (12) and \( n \mathcal{C}_k \) is the cardinality of \( \mathcal{C}_k \). To initialize, we set \( C_0 := \{v_i, \emptyset, \frac{1}{|\mathcal{V}_{s}|} | i = 1, \ldots, |\mathcal{V}_{s}| \} \) because each vertex in \( \mathcal{V}_{s} \) is equally likely to be the path starting vertex. The computational complexity of calculating each term in (12) is \( O(1) \) using the alias sampling method [37]. The upper bound of candidate vertex cardinality is \( |\mathcal{V}_{s}| \) and thus it takes \( O(|\mathcal{V}_{s}|) \) 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. [LV-C6]. With little variation in headings (e.g. Manhattan streets), \( |\mathcal{V}_{s}| \) is larger. On the contrary, \( |\mathcal{V}_{s}| \) is small compared to \( |\mathcal{V}_{s}| \) with large variation in road headings. In this case, \( O(|\mathcal{V}_{s}|) = O(1) \). The classification of probabilities into two groups is \( O(|\mathcal{V}_{s}|) \) using Hoare’s selection algorithm.

We summarize the computational complexity of Algorithm [1] in Lemma [1].

![Fig. 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.](image-url)
Lemma 1. The computation complexity of the heading-length matching is $O(n|V_h||V_{b,h}|)$.

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 $\mu_{d,k} = \mu_{d}$, $B_k = 1$ if $\mu_{\theta,k} = \mu_{\theta}$, 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 $M := \langle \Theta, D \rangle$ represent the true trajectory, whereas we know $A_1 \cdots A_n B_1 \cdots B_n$ from sequence matching. The analysis, we denote $n_v = |V_h|$ as the cardinality of $V_h$ and $n_b$ as the expected number of neighbors for each vertex. We describe map/trajecory 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 $n_b \gg k_l \geq n_b$ for most maps, we have the following lemma.

Lemma 2. The conditional probability that $M = \langle \Theta, D \rangle$ is the true matching sequence given that $Q = \langle \Theta_q, D_q \rangle$ matches $M$ is,

$$P(C_1 \cdots C_n | A_1 \cdots A_n B_1 \cdots B_n) = \left(\frac{1 - \alpha^2 k_d h}{n_v}\right) \left(1 - \alpha^2 \frac{k_d h}{n_b}\right)^{n-1} \tag{18}$$

Proof. Applying the Bayesian equation, we have

$$P(C_1 \cdots C_n | A_1 \cdots A_n B_1 \cdots B_n) = \frac{P(A_1 \cdots A_n B_1 \cdots B_n | C_1 \cdots C_n) P(C_1 \cdots C_n)}{P(A_1 \cdots A_n B_1 \cdots B_n)}. \tag{19}$$

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 $[16]$. Due to independent measurement noises, we have

$$P(A_1 B_1 | C_1) = (1 - \alpha)^2. \quad \text{Besides, these tests are independent due to independent sensor noises, we have}$$

$$P(A_1 \cdots A_n B_1 \cdots B_n | C_1 \cdots C_n) = (1 - \alpha)^{2n}. \tag{20}$$

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,

$$P(C_1 \cdots C_n) = \frac{1}{n_v!} \frac{1}{n_b!} n_v \cdot n_b. \tag{21}$$

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 $P(A_1 B_k) = 1/k_d k_l$, and

$$P(A_1 \cdots A_n B_1 \cdots B_n) = \left(\frac{1}{k_d k_l}\right)^n \tag{22}$$

Plugging (20), (21), and (22) into (19), we obtain the lemma. □

Corollary 1. We have shown in [17] that the conditional probability that $\Theta$ is the true matching given $\Theta_q$ is

$$P(C_1 \cdots C_n | B_1 \cdots B_n) = \left(\frac{1 - \alpha^2 k_d h}{n_v}\right) \left(1 - \alpha^2 \frac{k_d h}{n_b}\right)^{n-1} \tag{23}$$

Compare (18) with (23), we have

$$P(C_1 \cdots C_n | A_1 \cdots A_n B_1 \cdots B_n) = [(1 - \alpha)k_l] \tag{24}$$

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

According to (18), $(1 - \alpha)^2 \frac{k_d h}{n_b}$ determines localization efficiency which is related to both $k_d$ and $k_l$, the spreading of both heading and road length. To better understand how it stands in real world, we analyze map proprieties in the following section.

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 [38]. To minimize the effect of bin size on calculated entropy, we set orientation bin widths to be $5^\circ$, and 20 meters for road length. Let us denote orientation range set by $\{O_j | j = 1, 2, \cdots, n_j\}$ and length range set by $\{L_i | i = 1, 2, \cdots, n_i\}$. We define $n_{ji} = n_{ij} n_{i}$ and $p_{ji}$ be the relative frequency that $\theta_i \in O_j$ and $d_j \in L_i$. The joint Shannon entropy in heading and road length is

$$H_{\theta, d}(V_{h,l}) = -\sum_j \sum_i p_{ji} \log n_{ji} p_{ji}. \tag{25}$$

By analyzing the entropy of different maps, we predict localization efficiency of our algorithm, which will be shown in Section [V].
D. Location Alignment and Verification Thread

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 M_p (SSPTM), we can reset EKF parameters which rectifies the drifting issue.

Let us define the set of points in SSPT by

\[ X_q = \{ p_i \in \mathbb{R}^2 | i = 1, \cdots, n_q \} \]  

with each element obtained from EKF \( p_{i:2}^l = [x, y]^T \) where \( p_{i:2}^l \) is the first and second element of \( p_i^l \). The distribution of \( p_i \) is \( p_i \sim \mathcal{N}(\mu_p, \Sigma_p) \), where \( \mu_p \) is the mean vector and \( \Sigma_p \) is the covariance matrix obtained from the EKF. The corresponding GPS SSPT points are defined by

\[ X_h = \{ x_i | i = 1, \cdots, n_h \} \]  

and the covariance of GPS points is denoted by \( \Sigma_q = \text{diag}(\sigma_x^2, \sigma_y^2) \) as mentioned in Section IV-A. Thus we have \( x_i \sim \mathcal{N}(\mu_x, \Sigma) \).

1) Virtual Starting-Point and End-Point Estimation: However, SSPT points do not necessarily follow SSPTM as shown in Fig. 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. 7(a) also shows the effect of vehicle turn radius which makes the length of SSPTM 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 SSPTM segments. With the current segment positions \( X_q \), we denote the set of points from previous and next SSPTM segments by \( X_{q-} \) and \( X_{q+} \), respectively. Applying line fitting to \( X_q, X_{q-}, \) and \( X_{q+} \), we obtain three 2D lines \( L_q, L_{q-}, \) and \( L_{q+} \). Also, the line direction vectors are \( v_q = b_q - a_q \), \( v_{q-} = b_q - a_q \), and \( v_{q+} = b_q - a_q \). Finding the intersection between \( L_q \) and \( L_{q-} \) allows us to obtain the virtual starting point. We denote the virtual starting point of \( X_q \) by \( p_s \).

\[ p_s = a_q - \frac{v_q^\perp (a_q - a_q)}{v_{q-}^\perp \cdot v_q}, \]  

where \( \cdot \) is dot product and \( v_q^\perp \) is the perp operator of \( v_q \). Similarly, the intersection between \( L_q \) and \( L_{q+} \) gives us the virtual end point \( p_e \). We have

\[ p_e = a_q - \frac{v_q^\perp (a_q - a_q)}{v_{q+}^\perp \cdot v_q}, \]  

where \( v_{q+}^\perp \) is the perp operator of \( v_{q+} \). When SSPT is connected with an curve segment (e.g. caused by vehicle turn), we add \( p_s \) and \( p_e \) to \( X_q \) to help alignment process. \( p_s \) and \( p_e \) become the first and the last points in \( X_q \), respectively.

2) Location Alignment and Verification: With augmented \( X_q \), we can match \( X_q \) to \( X_h \) to rectify drifting issue by finding the transformation \( T \) between them (see Fig. 8). Here \( T \) is 3-DoF rigid body transformation represented by a 2x2 rotation matrix \( \mathbf{R} \), and a 2x1 translation vector \( \mathbf{t} \).

\[ T(x) := \mathbf{R}x + \mathbf{t}, \]

where \( x \) is a 2D point. \( X_q \) usually contains significantly more entries than that of \( X_h \) 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 \( X_h \) and minimizing the distance of all points in \( X_q \) to this line (Fig. 7(b)).

\[ d_p(T(p_s), L_h) = \frac{||a_h - T(p_s) \times (a_h - b_h)||}{||a_h - b_h||}, \]  

where ‘\( \times \)’ is the cross product and \( || \cdot || \) is the \( L^2 \) norm. We define the cost function \( C_T \) by

\[ C_T = \begin{bmatrix} d_p(T(p_s), L_h) \\ d_p(T(p_1), L_h) \\ \vdots \\ d_p(T(p_{n-1}), L_h) \\ d_p(T(p_n), L_h) \end{bmatrix}, \]
and formulate the following optimization problem

$$\arg \min \limits_T C_T^T \Sigma_C^{-1} C_T + \lambda \|T(p_r) - x_1\| + \lambda \|T(p_r) - x_{n_h}\|,$$  
(33)

where $\Sigma_C = \text{diag}(\sigma_2^2, \ldots, \sigma_p^2)$. $\beta$ is a nonnegative weight, and $x_1$ and $x_{n_h}$ are the first and the last entries in $\Sigma_\alpha$. $\sigma_2^2$ is obtained using error propagation. In detail, let $d_v(\mathbf{T}(p_r), \mathbf{L}_h) = f_d(p_r, \mathbf{L}_h)$ and $\xi = [p_r^T, \mathbf{L}_h^T]^T$, we have $\sigma_2^2 = J_q \Sigma_q J_q^T$, where $J_q = \frac{\partial f_d}{\partial x}$ and $\Sigma_q = \text{diag}(\Sigma_{p_r}, \Sigma_{\mathbf{L}_h})$ because $p_r$ is independent of $\mathbf{L}_h$ which comes from $X_h$. Define $\mathbf{L}_h = f_x(X_h)$, we have $\Sigma_{\mathbf{L}_h} = J_q \Sigma_q J_q^T$ where $J_q = \frac{\partial f_d}{\partial x}$ and $\Sigma_{\mathbf{L}_h} = \text{diag}(\Sigma_{\mathbf{L}_h}, \ldots, \Sigma_{\mathbf{L}_h})$. The second and third terms are soft constraints due to potential alignment errors. To solve (33), we start with a small positive weight for $\lambda$ and apply a nonlinear optimization solver, e.g., Levenberg-Marquardt algorithm. Initially, we set $R = I_{2 \times 2}$, and $t$ from the result of the global location obtained from Section IV-C. For each turn, we use previous solution as the initial solution and increase $\lambda$ gradually until the change in solution is negligible.

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

- $H_0$: $X_h$ and $\hat{\mathbf{X}}_q$ are from the same distribution,
- $H_1$: otherwise.  
(34)

We set the significance level by $\alpha$ and reject $H_0$ if the statistic is less than $\alpha$. Note $H_0$ is examined by the Mahalanobis distance $C_T^T \Sigma_C^{-1} C_T$ which follows a $\chi^2$ distribution with $2(n_q + 2)$ DoFs. Thus we reject $H_0$ if

$$C_T^T \Sigma_C^{-1} C_T > \chi^2_{2(n_q + 2)}(\alpha).$$

Correspondingly, we set localization status indicator variable $I_G$ values by

$$I_G = \left\{ \begin{array}{ll}
0, & H_0 \text{ is rejected,} \\
1, & \text{otherwise.}
\end{array} \right. $$
(35)

If $I_G = 1$, we accept $\mathbf{T}$ and use the aligned trajectory $\hat{\mathbf{X}}_q := \mathbf{T}(X_q)$ which is used to reset the EKF states (Fig. 2). After LAV execution, we keep acquiring the vehicle locations EKF $p_{1:2}$ 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.

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 [39]. 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 (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 $p_0$ and $p_n$ in (28) and (29). That is,

$$d_q = \|p_e - p_s\|.$$  
(36)

According to (27), the corresponding travel length on the map is denoted by $d := \|x_{n_h} - x_1\|$. Assuming GL thread ends at the $n$-th turn, for $k = (n+1), \ldots, n'$ we estimate $s_{ssf}$ by computing the ratio of accumulated length $d_{q,k}$ and $d_k$:

$$s_{ssf} = \sum_{k=n+1}^{n'} \frac{d_k}{d_{q,k}}.$$  
(37)

We then model the variance of $s_{ssf}$ to be used in the EKF measurement variance in Section IV-B1. It is not accurate to set a constant variance value for $s_{ssf}$, since at the beginning traveling length is short and thus $s$ has larger variance. As the traveling length increases, the variance of $s_{ssf}$ ought to decrease. Denote the variance of $s_{ssf}$ by $\sigma^2_{ssf}$; we derive the following Lemma.

**Lemma 3.** The variance of scale and slip factor $s_{ssf}$ is

$$\sigma^2_{ssf} = \frac{1}{L_q^2} (2n_q \sigma^2_g + \frac{L_q^2}{L_q^2} \sum_{k=n+1}^{n'} \sigma^2_{d_{q,k}}).$$  
(38)

**Proof.** If we write $s_{ssf}$ as function of measurements from $d_k$ and $d_{q,k}$ according to (37). Then, $\sigma^2_{ssf} = \sigma^2_{d_{q,k}}$ from (5) and the variance of $d_{q,k}$ which is defined in Section IV-B2. Let us define $L_q = \sum_{k=n+1}^{n'} d_{q,k}$, $L_q = \sum_{k=n+1}^{n'} d_k$, and $n_q = n - n$. Through forward error propagation,

$$\sigma^2_{ssf} = J_q \Sigma_q J_q^T,$$  
(39)

where $\Sigma_q = \text{diag}(2 \sigma^2_g, \ldots, 2 \sigma^2_g, \sigma^2_{d_{q,n+1}}, \ldots, \sigma^2_{d_{q,n'}})$ and $J_q$ is

$$J_q = [\frac{\partial f_s}{\partial d_{n+1}}, \ldots, \frac{\partial f_s}{\partial d_{q,n+1}}, \ldots, \frac{\partial f_s}{\partial d_{q,n'}}] = \left[ \begin{array}{c}
\frac{1}{L_{q}}, \\
\frac{1}{L_{q}}, \\
\ldots, \\
\frac{1}{L_{q}}, \\
\frac{L_q}{L_{q}}, \\
\frac{L_q}{L_{q}}, \\
\ldots, \\
\frac{L_q}{L_{q}}, \\
\frac{L_q}{L_{q}}.
\end{array} \right].$$  
(40)

Plug (40) into (39), we have

$$\sigma^2_{ssf} = \frac{L_q}{L_{q}^2} \sum_{k=n+1}^{n'} \frac{L_{q}^2}{L_{q}^2} \sigma^2_{d_{q,k}}.$$  
(41)

**Remark 1.** Let us take a close look at (41). We have $L_q \approx L_q$ because the estimated travel length should be similar to the corresponding path in map. Therefore, we can approximate $\sigma^2_{ssf}$ as

$$\sigma^2_{ssf} = J_q \Sigma_q J_q^T = \frac{1}{L_q^2} (2n_q \sigma^2_g + \sum_{k=n+1}^{n'} \sigma^2_{d_{q,k}}).$$

Thus we show that $\sigma^2_{ssf}$ 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 compared in our experiments: important measure for algorithm efficiency. Two algorithms is uniquely localized. The number of solutions is also an map. If the number of solutions is one, then the vehicle

metric for this. For a given
inputs it takes to localize the vehicle. Therefore, computation
is a good

\[ \sigma_{\text{ssf}}^2 \]

\[ \sigma_{\text{v}}^2 \]

are fed into the EKF in Section [IV-B1]. This completes our overall method.

V. 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. I.

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 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.

A. Global Localization Test

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 [I].

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. [2]. 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. 9(a), we calculate map entropy distributions of 100 cites based on the data from [41]. 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. 9(b) and Fig. 9(c) As shown in Fig. 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 [I], 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 \( \sigma_{\theta_k} = 5^\circ \), \( \sigma_{d_{k}} = \sqrt{2}\sigma_{g} \), 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. 9(d) and we adapt Fig. 9(d) from [I] for comparison.

For PLAM which uses heading only (Fig. 9(d)), the vehicle can be localized with \( n \leq 10 \) if the entropy in orientation is above 0.9 [I]. 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. 9(d) and Fig. 9(e) 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. 9(f) and 9(g) 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. [I].

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. I. In all tests, GBPL takes \( n = 3.1 \)

| MAP INFO. AND #STRAIGHT SEGMENTS n FOR LOCALIZATION |
|---------------------------------|--------------|-------------|---|---|---|
| Maps | Size (arc) | Drivable road (km) | Entropy | #nodes | n (PLAM) | n (GBPL) |
| CSMap | 3.24 | 35.77 | 0.924 | 438 | 9.5 | 3.2 |
| KITTI00Map | 4.75 | 44.2 | 0.877 | 583 | 10.5 | 4.5 |
| KITTI05Map | 3.24 | 43.7 | 0.797 | 548 | 4.5 | 3.4 |
B. 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.

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.

![Graphs](image)

Fig. 10. LAV accuracy results using KITTI sequences on KITTI00Map and KITTI05Map: (a) KITTI00-1, (b) KITTI00-2, (c) KITTI05-1, and (d) KITTI05-2.

2) Accuracy Results: Figs. [10] and [11] show the accuracy results by plotting the localization errors of each sequence. Red vertical lines are where LAV is executed, 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 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.

3) Scale and Slip Factor: Fig. [12] shows the estimated SSF in EKF (i.e., $s_j$ in [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

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. [I] 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. [9(e)] $n$ required for reaching a unique solution in the real map agrees with simulation results.
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. 13. These results show $\sigma^2_{s_j}$ decreasing as travel length increases as in Lemma [5].

VI. CONCLUSION AND FUTURE WORK

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 rudimentary 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.

In the future, we are interested in extending the work to design a multiple vehicle/robot collaborative localization scheme under ad hoc vehicle-to-vehicle communication framework. We will report new results in the future publications.

ACKNOWLEDGMENT

We would like to thank C. Chou, B. Li, S. Yeh, A. Kingery, A. Angert, D. Wang, and S. Xie for their input and contributions to the NetBot Lab at Texas A&M University.

REFERENCES

[1] H. Cheng, D. Song, A. Angert, B. Li, and J. Yi, “Proprioceptive localization assisted by magnetoreception: A minimalistic intermittent heading-based approach,” IEEE Robotics and Automation Letters, 2018.
[2] M. A. Brubaker, A. Geiger, and R. Urtasun, “Map-based probabilistic visual self-localization,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 4, pp. 652–665, April 2016.
[3] S. Lowry, N. Sunderhauf, 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.
[4] Y. Lu and D. Song, “Visual navigation using heterogeneous landmarks and unsupervised geometric constraints,” in IEEE Transactions on Robotics (T-RO), vol. 31, no. 3, June 2015, pp. 736–749.
[5] 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, 2003, pp. 206–211.
[6] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2d lidar slam,” in Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, 2016, pp. 1271–1278.
[7] J. Levinson and S. Thrun, “Robust vehicle localization in urban environments using probabilistic maps,” in Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010, pp. 4372–4378.
[8] 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.
[9] 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.
[10] H. Aly, A. Basalamah, and M. Yousef, “Accurate and energy-efficient gps-less outdoor localization,” ACM Trans. Spatial Algorithms Syst., vol. 3, no. 2, pp. 4:1–4:31, Jul. 2017.
[11] C. Chou, A. Kingerly, D. Wang, H. Li, and D. Song, “Encoder-camera-ground penetrating radar tri-sensor mapping for surface and subsurface transportation infrastructure inspection,” in 2018 IEEE International Conference on Robotics and Automation (ICRA), May 2018, pp. 1452–1457.
[12] 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.
[13] J. Yi, J. Zhang, D. Song, and S. Jayasuriya, “imu-based localization and slip estimation for skid-steered mobile robots,” in Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on. IEEE, 2007, pp. 2845–2850.
[14] 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.
[15] 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.
[16] I. Constandache, R. R. Choudhury, and I. Rhee, “Compacc: Using mobile phone compasses and accelerometers for localization,” in IEEE INFOCOM. Citeseer, 2010, pp. 1–9.
[17] 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). IEEE, 2010, pp. 4868–4873.
[18] 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.
[19] L. Huang, B. He, and T. Zhang, “An autonomous navigation algorithm for underwater vehicles based on inertial measurement units and sonar,” in 2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010), vol. 1. IEEE, 2010, pp. 311–314.
[20] B. Siciliano and O. Khatib, Springer handbook of robotics. Springer, 2016.
[21] P. Merriaux, Y. Dupuis, P. Vasseur, and X. Savatier, “Fast and robust vehicle positioning on graph-based representation of drivable maps,” in Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, 2015, pp. 2787–2793.
[22] P. Ruchti, B. Steder, M. Ruhnke, and W. Burgard, “Localization on opentreetmap data using a 3d laser scanner,” in Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, 2015, pp. 5260–5265.
[23] 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.
[24] Y. Jin and Z. Xiang, “Robust localization via turning point filtering with road map,” in Intelligent Vehicles Symposium (IV), 2016 IEEE. IEEE, 2016, pp. 992–997.
[25] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. MIT Press, 2005.
[26] OpenStreetMap contributors, “Planet dump retrieved from https://planet.osm.org,” 2017.
[27] Google Maps contributors, “https://www.google.com/maps/” 2017.
[28] M. Quddus and S. Washington, “Shortest path and vehicle trajectory aided map-matching for low frequency gpg data,” Transportation Research Part C: Emerging Technologies, vol. 55, pp. 328–339, 2015.
[29] 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), 2011, pp. 2564–2571.
[30] J. Wahlstrm, I. Skog, J. G. P. Rodrigues, P. Hndel, and A. Aguiar, “Map-aided dead-reckoning using only measurements of speed,” IEEE Transactions on Intelligent Vehicles, vol. 1, no. 3, pp. 244–253, Sep. 2016.
[31] 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.
[32] R. Hartley and A. Zisserman, Multiple view geometry in computer vision. Cambridge university press, 2003.
[33] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, Estimation with applications to tracking and navigation: theory algorithms and software. John Wiley & Sons, 2004.
[34] 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, no. 5, pp. 1087–1097, Oct 2009.
[35] H.-M. Cheng and D. Song, “Localization in inconsistent wif environments,” in The International Symposium on Robotics Research (ISRR), Puerto Varas, Chile, 2017.
[36] 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.
[37] 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.
[38] N. Mohajeri and A. Gudmundsson, “The evolution and complexity of urban street networks,” Geographical Analysis, vol. 46, no. 4, pp. 345–367, 2014.
[39] J. Borenstein and L. Feng, “Correction of systematic odometry errors in mobile robots,” in Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots, vol. 3. IEEE, 1995, pp. 569–574.
[40] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012, pp. 3354–3361.
[41] G. Boeing, “Urban spatial order: Street network orientation, configuration, and entropy,” Applied Network Science, vol. 4, no. 1, p. 67, 2019.