This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

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

Hsin-Min Cheng and Dezhen Song, *Senior Member, IEEE*

***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 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, 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 preprocessed 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 10 m).**

***Index Terms*—Autonomous vehicle navigation, localization, sensor fusion.**

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 (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).

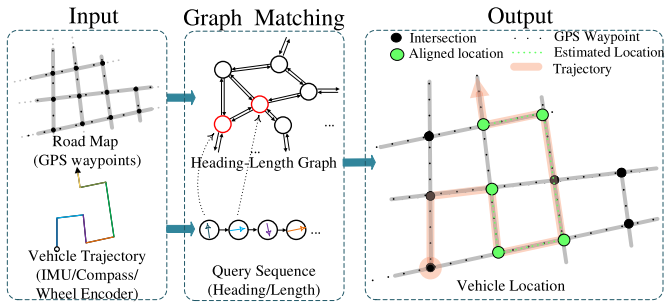


Fig. 1. Illustration of GBPL method. Left: Our inputs include a prior known map and the trajectory estimated from an IMU, a compass, and a wheel encoder. Middle: We process the prior map in to a straight segment connectivity graph and also the trajecory into a query sequence of headings and lengths of straight segments. Right: Aligned trajectory to the map after graph matching.

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 nontrivial 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 (see 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 preprocess 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.

Manuscript received September 8, 2020; accepted December 15, 2020. This work was supported by National Science Foundation under Grant NRI-1925037. This paper was recommended for publication by Associate Editor S. Huang and Editor F. Chaumette upon evaluation of the reviewers' comments. *(Corresponding author: Dezhen Song.)*

The authors are with the CSE Department, Texas A&M University, College Station, TX 77843 USA (e-mail: hmcheng@tamu.edu; dzsong@cs.tamu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2020.3046419.

Digital Object Identifier 10.1109/TRO.2020.3046419

1552-3098 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

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 10 m).

The rest of the article 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. Section VI concludes this article.

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 proprioceptive 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^{TM} (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 HLG 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 m. In [21], the velocity from wheel encoder and steering angles are used for odometry and a PF-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 m, 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 article 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 (LAV) after graph matching.

III. Problem Formulation

A. Scenarios and Assumptions

In our setup, 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 onboard 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.

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.

Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

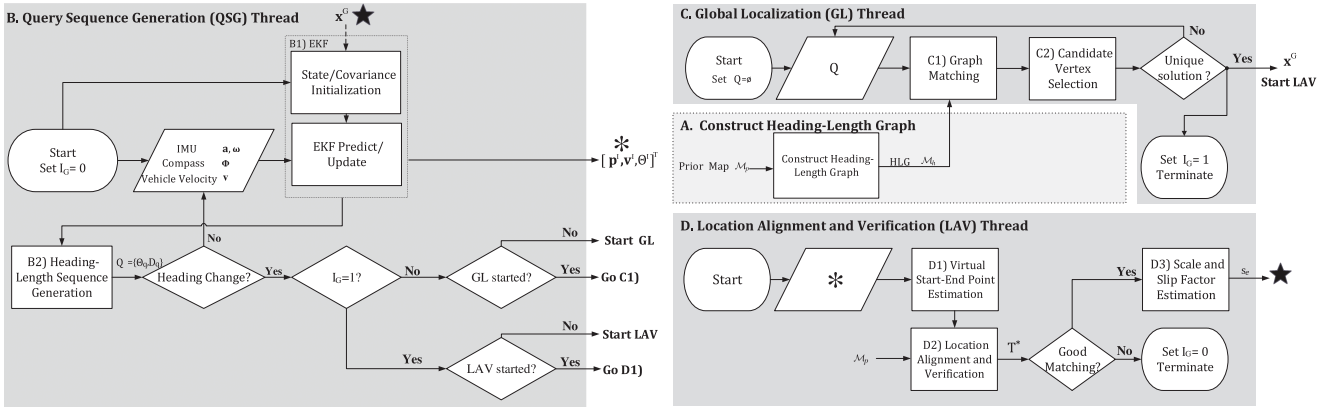


Fig. 2. System diagram.

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 colocated, precalibrated, 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 20 m.

B. Nomenclature

Common notations are defined as follows.

1) $\mathcal{M}_p := \{\mathbf{x}_m = [x_m, y_m]^\mathsf{T} \in \mathbb{R}^2 | m \in \mathscr{M}\}$ 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.
2) $\mathbf{a} = \{\mathbf{a}_j \in \mathbb{R}^3 | j = 0, 1, \ldots, N_j\}$ and $\omega = \{\boldsymbol{\omega}_j \in \mathbb{R}^3 | j = 0, 1, \ldots, N_j\}$ denote accelerometer readings and gyroscope angular velocities from the IMU, respectively.
3) $\phi = \{\phi_{j_\phi} \in \mathbb{R} | j_\phi = 0, \ldots, \lfloor \frac{N_j}{c_\phi} \rfloor\}$ denotes compass readings where $c_\phi \geq 1$ since a compass often has lower sampling frequency than that of the IMU.
4) $\mathbf{v} = \{v_{j_v} \in \mathbb{R} | j_v = 0, \ldots, \lfloor \frac{N_j}{c_v} \rfloor\}$ 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.
5) $\mathcal{M}_h = \{\mathcal{V}_h, \mathcal{E}_h\}$ denotes the HLG where $\mathcal{V}_h$ is the vertex set and $\mathcal{E}_h$ and is the edge set.
6) $Q = \{\Theta_q, D_q\}$ 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.
7) $\mathcal{C}_k$ represents the candidate vertex set where $k = 1, \ldots, n$ is the length of the query sequence.

The GBPL problem is defined as follows.

Problem 1: 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.

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 multithreaded 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.

A. HLG Construction

We preprocess 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.

1) 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. 3 shows an example. For the same route, the trajectories may be different due to driving on different lanes, driver habit, traffic, etc.

Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

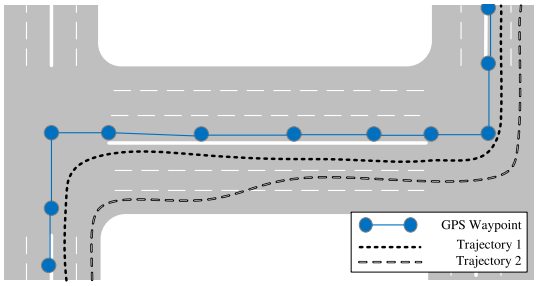


Fig. 3. Map and trajectory discrepancy illustration. Given the trajectory generated by proprioceptive sensors, directly matching trajectory with the map may not be desirable. For the same route, trajectories 1 and 2 appear quite differently. Neither of them matches blue waypoints in the map.

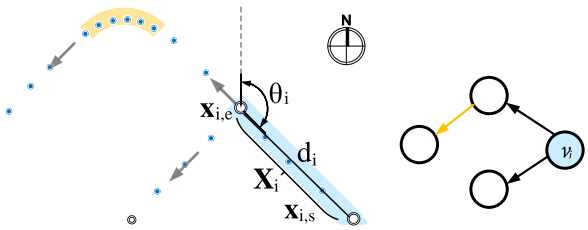


Fig. 4. HLG illustration in color. The left figure shows a satellite image with road map consisted of GPS waypoints (blue dots) overlaying on top of the image and intersections represented in small black circles. We estimate road curvature changes to capture heading change and construct HLG. As an example, 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.

2) 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 $\mathcal{M}_p$.
3) 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 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]. 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:

$$v_i = \{\mathbf{X}_i, \theta_i, d_i, b_i\} \tag{1}$$

where $\mathbf{X}_i = [\mathbf{x}_{i,s}^\mathsf{T}, \ldots, \mathbf{x}_{i,e}^\mathsf{T}]^\mathsf{T}$ contained all 2-D 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

$$d_i = ||\mathbf{x}_{i,s} - \mathbf{x}_{i,e}|| \tag{2}$$

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

$$b_i = \begin{cases} 1, & d_i > t_l \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

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 [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 σ_g^2. According to [2], typical consumer grade navigation systems offer positional accuracy around $\sigma_g = 10$ m. The distribution of θ_i that characterizes its uncertainty is

$$\theta_i \sim \mathcal{N}(\mu_{\theta_i}, \sigma_{\theta_i}^2) \tag{4}$$

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

$$d_i \sim \mathcal{N}(\mu_{d_i}, \sigma_{d_i}^2) = \mathcal{N}(\mu_{d_i}, 2\sigma_g^2). \tag{5}$$

B. 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: $\mathbf{a}$, $\boldsymbol{\omega}$, $\boldsymbol{\phi}$, and $\mathbf{v}$, 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 ϕ_0 to determine the initial

Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

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

$$\mathbf{X}_{s,j} := [\mathbf{p}_j^I, \mathbf{v}_j^I, \mathbf{\Theta}_j^I, s_j]^\mathsf{T} \tag{6}$$

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 $\mathbf{\Theta}^I := [\alpha, \beta, \gamma]^\mathsf{T}$ 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:

$$\begin{aligned} \mathbf{p}_j^I &= \mathbf{p}_{j-1}^I + \tau_\omega \mathbf{v}^I \\ \mathbf{v}_j^I &= \mathbf{v}_{j-1}^I + \tau_\omega ({}_B^I\mathbf{R}(\mathbf{a}) - \mathbf{G}) \\ \Theta_j &= \Theta_{j-1} + \tau_\omega {}_B^I\mathbf{E}(\boldsymbol{\omega}) + \mathbf{c}_\gamma \\ s_j &= s_{j-1} \end{aligned} \tag{7}$$

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, ${}_B^I\mathbf{R}$ is the rotation matrix from $\{B\}$ to $\{I\}$, and ${}_B^I\mathbf{E}$ 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 IV-D3. 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

$$({}_I^B\mathbf{R})_{2:3}\mathbf{v}_j^I = \begin{bmatrix} 0 & 0 \end{bmatrix}^\mathsf{T} \tag{8}$$

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

From the coordinate definition, the heading direction is γ defined in $\{I\}$ (last component of $\mathbf{\Theta}^I$), 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

$$\gamma_j = \begin{cases} \phi_{j_\phi}, & \text{if } j = c_\phi j_\phi \\ \phi_{j_\phi - 1}, & \text{otherwise.} \end{cases} \tag{9}$$

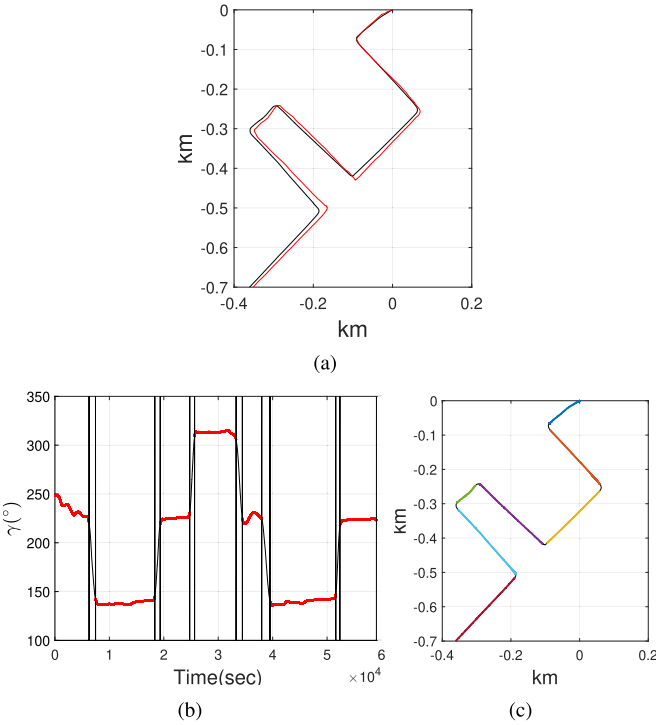


Fig. 5. (a) Trajectory estimation result: the red line is the GPS ground truth, and black line illustrates the EKF estimated trajectory. (b) Query heading representations. Blue line is estimated heading, black vertical lines are indices where data segmented, red lines mark out stable heading segments and unmarked segments are detected turns. (c) Corresponding travel heading and length segment representations. Different segments are marked in different colors.

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

$$s_j = s_{ssf} \tag{10}$$

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

$$||\mathbf{v}_j^I|| = \begin{cases} s_j v_{j_v}, & \text{if } j = c_v j_v \\ s_j v_{j_v - 1}, & \text{otherwise.} \end{cases} \tag{11}$$

Combining (9)–(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

Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

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. 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)].

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 its 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 if compare it 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.

C. GL Thread

1) GL Overview: With the query sequence obtained from onboard 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 article.

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.

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 $M := \{\Theta, D\} = \{(\theta_k, d_k) | k = 1, \ldots, n\}$ correspondingly. As a convention in this article, for random vector $\star$, $\mu_\star$ represents its mean vector. Following the convention, mean matrix of Q is defined as $\mu_Q = [\mu^\mathsf{T}_{\Theta_q}, \mu^\mathsf{T}_{D_q}]^\mathsf{T}$ where $\mu_{\Theta_q} = [\mu_{\theta_{q,1}}, \ldots, \mu_{\theta_{q,n}}]^\mathsf{T}$ and $\mu_{D_q} = [\mu_{d_{q,1}}, \ldots, \mu_{d_{q,n}}]^\mathsf{T}$. The mean matrix of M is denoted by $\mu_M = [\mu^\mathsf{T}_\Theta, \mu^\mathsf{T}_D]^\mathsf{T}$ where $\mu_\Theta = [\mu_{\theta_1}, \ldots, \mu_{\theta_n}]^\mathsf{T}$ and $\mu_D = [\mu_{d_1}, \ldots, \mu_{d_n}]^\mathsf{T}$.

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

$$\begin{aligned} &P(\mu_Q = \mu_M | Q, M) \\ &= P(\mu_{\Theta_q} = \mu_\Theta | \Theta_q, \Theta) P(\mu_{D_q} = \mu_D | D_q, D). \end{aligned} \tag{12}$$

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

$$P(\mu_{\Theta_q} = \mu_\Theta | \Theta_q, \Theta) \propto \prod_{k=1}^{n} f_T(t(\theta_{q,k}, \theta_k)) \tag{13}$$

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

$$P(\mu_{D_q} = \mu_D | D_q, D) \propto \prod_{k=1}^{n} f(z(d_{q,k}, d_k)) \tag{14}$$

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 (13) and (14) and recalling that n is the number of straight segments in the query sequence, we rewrite (12) as follows:

$$P(\mu_Q = \mu_M | Q, M) \propto \prod_{k=1}^{n} f_T(t(\theta_{q,k}, \theta_k)) f(z(d_{q,k} - d_k)). \tag{15}$$

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

$$\begin{aligned} \mathbf{H_0}: &\quad [\mu_{\theta_{q,k}}, \mu_{d_{q,k}}]^\mathsf{T} = [\mu_{\theta_k}, \mu_{d_k}]^\mathsf{T} \\ \mathbf{H_1}: &\quad \text{otherwise.} \end{aligned} \tag{16}$$

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

$$(|t(\theta_k, \theta_{q,k})| > t_{\alpha/2,\nu}) \vee (|z(d_k, d_{q,k})| > z_{\alpha/2}). \tag{17}$$

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

Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

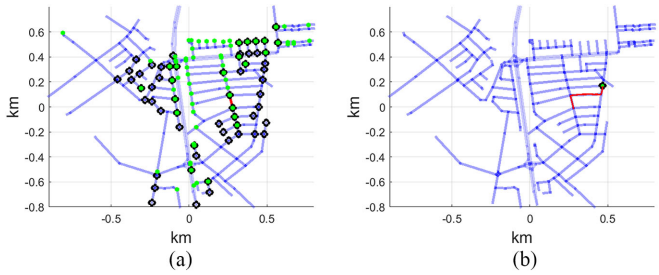


Fig. 6. Example of GL. (a) Candidate locations using heading matching (green dots), length matching (black circle). We show that performing heading-length matching (locations with green dot and black circle) helps reducing candidates. (b) Candidate localization is reduced to the single solution if the joint distribution between heading and length is used.

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 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 (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 $k = 1, \cdots, n$ is the length of the query sequence. The candidate vertex set is denoted by $\mathcal{C}_k = \{\{v_i, \mathcal{V}_{M,i}, p_i\} | i = 1, \ldots, n_{\mathcal{C}_k}\}$, 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 (12) and $n_{\mathcal{C}_k}$ is the cardinality of $\mathcal{C}_k$. To initialize, we set $\mathcal{C}_0 := \{\{v_i, \emptyset, \frac{1}{|\mathcal{V}_{h,l}|}\} | i = 1, \ldots, |\mathcal{V}_{h,l}|\}$ 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 (12) is $O(1)$ using the alias sampling method [37]. 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 Section IV-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 $|\mathcal{V}_{h,l}|$ 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.

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	$\mathcal{C}_0 := \{\{v_i, \emptyset, \frac{1}{\lvert\mathcal{V}_{h,l}\rvert}\} \lvert i = 1, \cdots, \lvert\mathcal{V}_{h,l}\rvert\}$	$O(1)$
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** $i = 1, \cdots, n_{\mathcal{C}_{k-1}}$ **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* (16) **then**	
16	$\quad\quad\quad\quad$ Update matching probability $p_{i'} \leftarrow p_i \cdot p$	$O(1)$
17	$\quad\quad\quad\quad \mathcal{V}_{M,i'} \leftarrow$ 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 probabilities in (12) of $\mathcal{C}_k$ using Otsu's method;	$O(\lvert\mathcal{V}_s\rvert\lvert\mathcal{V}_{h,l}\rvert)$
20	$\quad$ Remove group in $\mathcal{C}_k$ with lower probabilities;	$O(1)$
21	$\quad$ **if** $\lvert\mathcal{C}_k\rvert > 1$ **then**	
22	$\quad\quad$ Return $\mathcal{C}_k$;	$O(1)$
23	$\quad$ **else**	
24	$\quad\quad$ Set $I_G = 1$;	$O(1)$
25	$\quad\quad$ Return vehicle location;	$O(1)$

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

Lemma 1: The computation complexity of the heading-length matching is $O(n|\mathcal{V}_s||\mathcal{V}_{h,l}|)$.

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.

Lemma 2: The conditional probability that $M = \{\Theta, D\}$ is the true matching sequence given that $Q = \{\Theta_q, D_q\}$ matches M is

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

Proof: Applying the Bayesian equation, we have

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

Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

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 $P(A_1 B_1 | C_1) = (1-\alpha)^2$. 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

$$P(C_1 \cdots C_n) = \frac{1}{n_b^{n-1}} \frac{1}{n_v}. \tag{21}$$

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 $P(A_k B_k) = \frac{1}{k_d k_l}$ and

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

Plugging (20)–(22) into (19), 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

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

Compare (18) with (23), we have

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

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 2 shows the probabilistic convergence of right matching. Also, it reveals when the localization scheme works and localization efficiency. If $P(C_1 C_2 \cdots C_n | A_1 A_2 \cdots A_n B_1 B_2 \cdots B_n)$ increases as n increases, the proposed method would find the correct location eventually. The localization speed is determined by $(1-\alpha)^2 \frac{k_d k_l}{n_b}$ 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 $P(C_1 C_2 \cdots C_n | A_1 A_2 \cdots A_n B_1 B_2 \cdots B_n)$. If a map contains many different road headings and lengths, then $P(C_1 C_2 \cdots C_n | A_1 A_2 \cdots A_n B_1 B_2 \cdots B_n)$ 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 $P(C_1 C_2 \cdots C_n | A_1 A_2 \cdots A_n B_1 B_2 \cdots B_n)$ and the algorithm fails. Fortunately, most maps do not have the issue [38]. 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.

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 [39]. To minimize the effect of bin size on calculated entropy, we set orientation bin widths to be 5°, and 20 m for road length. Let us denote orientation range set by $\{O_{\mathtt{j}} | \mathtt{j} = 1, 2, \ldots, n_{\mathtt{j}}\}$ and length range set by $\{L_{\mathtt{i}} | \mathtt{i} = 1, 2, \ldots, n_{\mathtt{i}}\}$. We define $n_{ji} = n_j n_i$ 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

$$H_{\theta,d}(\mathcal{V}_{h,l}) = -\sum_{\mathtt{j}} \sum_{\mathtt{i}} \rho_{\mathtt{ji}} \log_{n_{ji}} \rho_{\mathtt{ji}}. \tag{25}$$

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

D. LAV 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 $\mathcal{M}_p$ (SSPTM), we can reset EKF parameters which rectifies the drifting issue.

Let us define the set of points in SSPTE by

$$\mathbf{X}_q = \{\mathbf{p}_\iota \in \mathbb{R}^2 | \iota = 1, \ldots, n_q\} \tag{26}$$

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

$$\mathbf{X}_h = \{\mathbf{x}_l | l = 1, \ldots, n_h\} \tag{27}$$

and the covariance of GPS points is denoted by $\Sigma_g = diag(\sigma_g^2, \sigma_g^2)$ as mentioned in Section IV-A. 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 $\mathbf{x}_l \sim \mathcal{N}(\mu_{\mathbf{x}_l}, \Sigma_g)$.

1) Virtual Starting-Point and End-Point Estimation: However, SSPTE points do not necessary 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 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

Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

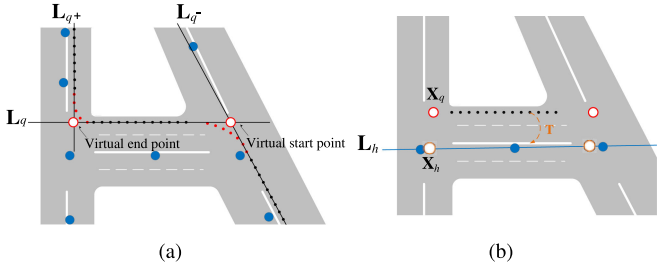


Fig. 7. Illustration of LAV. The solid small dots represent vehicle trajectory where red points are turn points and black points belong to SSPTE. The roads are shaded gray regions characterizing their width, and GPS waypoints in $\mathcal{M}_p$ are represented in larger blue dots. (a) Virtual starting 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.

$\mathbf{L}_q = [\mathbf{a}_q^\mathsf{T}, \mathbf{b}_q^\mathsf{T}]^\mathsf{T}$, $\mathbf{L}_{q^-} = [\mathbf{a}_{q^-}^\mathsf{T}, \mathbf{b}_{q^-}^\mathsf{T}]^\mathsf{T}$, 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 $\mathbf{p}_s$

$$\mathbf{p}_s = \mathbf{a}_q - \frac{\mathbf{v}_{q^-}^\perp.(\mathbf{a}_q - \mathbf{a}_q^-)}{\mathbf{v}_{q^-}^\perp.\mathbf{v}_q}\mathbf{v}_q \tag{28}$$

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

$$\mathbf{p}_e = \mathbf{a}_q - \frac{\mathbf{v}_{q^+}^\perp.(\mathbf{a}_q - \mathbf{a}_q^+)}{\mathbf{v}_{q^+}^\perp.\mathbf{v}_q}\mathbf{v}_q \tag{29}$$

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.

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. 8). Here $\mathbf{T}$ is 3-DoF rigid body transformation represented by a 2x2 rotation matrix $\mathbf{R}$, and a 2x1 translation vector $\mathbf{t}$

$$\mathbf{T}(\mathbf{x}) := \mathbf{R}\mathbf{x} + \mathbf{t} \tag{30}$$

where $\mathbf{x}$ is a 2-D 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 [see Fig. 7(b)].

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

$$d_\perp(\mathbf{T}(\mathbf{p}_\iota), \mathbf{L}_h) = \frac{||(\mathbf{a}_h - \mathbf{T}(\mathbf{p}_\iota) \times (\mathbf{a}_h - \mathbf{b}_h)||}{||\mathbf{a}_h - \mathbf{b}_h||} \tag{31}$$

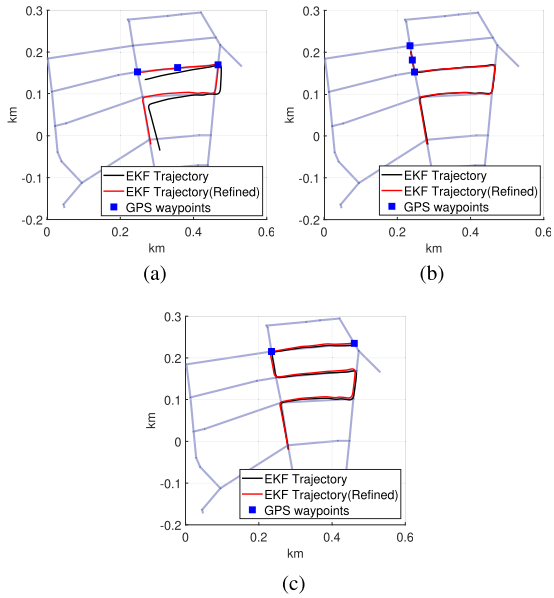


Fig. 8. Example of LAV that keeps drifting under control where n is the number of long straight segments for the vehicle. The unaligned trajectory is shown in black, the aligned trajectory is shown in red, and GPS waypoints are shown in dark blue square. (a) $n = 4$. (b) $n = 5$. (c) $n = 6$.

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

$$\mathbf{C_T} = \begin{bmatrix} d_\perp(\mathbf{T}(\mathbf{p}_s), \mathbf{L}_h) \\ d_\perp(\mathbf{T}(\mathbf{p}_1), \mathbf{L}_h) \\ \vdots \\ d_\perp(\mathbf{T}(\mathbf{p}_{n_q}), \mathbf{L}_h) \\ d_\perp(\mathbf{T}(\mathbf{p}_e), \mathbf{L}_h) \end{bmatrix} \tag{32}$$

and formulate the following optimization problem:

$$\arg\min_{\mathbf{T}} \mathbf{C}_\mathbf{T}^\mathsf{T}\Sigma_C^{-1}\mathbf{C_T} + \lambda||\mathbf{T}(\mathbf{p}_s) - \mathbf{x}_1|| + \lambda||\mathbf{T}(\mathbf{p}_e) - \mathbf{x}_{n_h}|| \tag{33}$$

where $\Sigma_C = diag(\sigma^2_{d_\perp,\mathbf{p}_s}, \ldots, \sigma^2_{d_\perp,\mathbf{p}_e})$, β is a nonnegative weight, and $\mathbf{x}_1$ and $\mathbf{x}_{n_h}$ are the first and the last entries in (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 (33), we start with a small positive weight for λ and apply a nonlinear optimization solver, e.g., Levenberg–Marquardt algorithm. Initially, we set $\mathbf{R} = \mathbf{I}_{2\times2}$, and $\mathbf{t}$ 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 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

Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

hypotheses

$$\mathbf{H_0}: \mathbf{X}_h \text{ and } \hat{\mathbf{X}}_q \text{ are from the same distribution}$$
$$\mathbf{H_1}: \text{otherwise.} \tag{34}$$

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 $\mathbf{C_T^T}\Sigma_C^{-1}\mathbf{C_T}$ which follows a χ^2 distribution with $2(n_q+2)$ DoFs. Thus, we reject H_0 if $\mathbf{C_T^T}\Sigma_C^{-1}\mathbf{C_T} > \chi^2_{2(n_q+2)}(\alpha)$. 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 (see Fig. 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.

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 nonsystematic biases from wheel encoder inputs [40]. The systematic error can be caused by tire radius error such as inflation level, tire wear, gear ratio, etc. Nonsystematic 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 $\mathbf{p}_e$ and $\mathbf{p}_s$ in (28) and (29). That is $d_q = ||\mathbf{p}_e - \mathbf{p}_s||$. According to (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 $k = (n+1), \ldots, n'$ we estimate s_{ssf} by computing the ratio of accumulated length $d_{q,k}$ and d_k

$$s_{ssf} = \sum_{k=n+1}^{n'} d_k \Big/ \sum_{k=n+1}^{n'} d_{q,k}. \tag{35}$$

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_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 3: The variance of scale and slip factor s_{ssf} is

$$\sigma^2_{s_{ssf}} = \frac{1}{L_q^2}(2n_s\sigma_g^2 + \frac{L_g^2}{L_q^2}\sum_{k=n+1}^{n'}\sigma^2_{d_q,k}). \tag{36}$$

Proof: First, we write s_{ssf} as function of measurements from d_k and $d_{q,k}$ according to (35). That is, $s_{ssf} = f_s(d_{n+1}, \ldots, d_{n'}, d_{q,n+1}, \ldots, d_{q,n'})$. We know the variance of d_k is $\sigma^2_{d_k} = 2\sigma_g^2$ from (5) and the variance of $d_{q,k}$ is $\sigma^2_{d_q,k}$ which is defined in Section IV-B2. 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

$$\sigma^2_{s_{ssf}} = J_s\Sigma_s J_s^{\mathrm{T}} \tag{37}$$

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

$$\begin{aligned} J_s &= \left[\frac{\partial f_s}{\partial d_{n+1}}, \ldots, \frac{\partial f_s}{\partial d_{n'}}, \frac{\partial f_s}{\partial d_{q,n+1}}, \ldots, \frac{\partial f_s}{\partial d_{q,n'}}\right] \\ &= \left[\frac{1}{L_q} \cdots, \frac{1}{L_q}, \frac{-L_g}{L_q^2}, \ldots, \frac{-L_g}{L_q^2}\right]. \end{aligned} \tag{38}$$

Plug (38) into (37), we have

$$\begin{aligned} \sigma^2_{s_{ssf}} = J_s\Sigma_s J_s^{\mathrm{T}} &= 2n_s\frac{\sigma_g^2}{L_q^2} + \sum_{k=n+1}^{n'}\sigma^2_{d_q,k}\frac{L_g^2}{L_q^4} \\ &= \frac{1}{L_q^2}\left(2n_s\sigma_g^2 + \frac{L_g^2}{L_q^2}\sum_{k=n+1}^{n'}\sigma^2_{d_q,k}\right). \end{aligned} \tag{39}$$

■

Remark 1: Let us take a close look at (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 $\sigma^2_{s_{ssf}}$ as

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

Thus we show that $\sigma^2_{s_{ssf}}$ 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 compensated, setting a sliding window for accumulated distance can be used to detect nonsystematic biases that varies through traveling.

The resulting s_{ssf} and $\sigma^2_{s_{ssf}}$ 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 GL approach. Second, we test the LAV performance.

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

1) CSMap: College Station, Texas, U.S.;
2) KITTI00Map: Karlsruhe, Germany;
3) 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 Table I .

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

Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
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**

Bold entities to emphasize the superiority in performance of the proposed method comparing with the state-of-the-art.

1) *CSData:* We record IMU readings at 400 Hz and compass readings at 50 Hz using a Google Pixel phone mounted on a passenger car. Also, we read the vehicle speed at 46.6 Hz 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.
2) *KITTI:* We use the KITTI GPS/IMU dataset [41] 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 datasets do not provide compass readings. We have four sequences: KITTI00-1, KITTI00-2, KITTI05-1, and KITTI05-2.

A. GL 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. The following two algorithms are compared in our experiments.

1) GBPL: Current method that uses both heading and length information of straight segments.
2) PLAM: The counterpart method using heading only [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 Lemma 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 [38]. 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 (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.

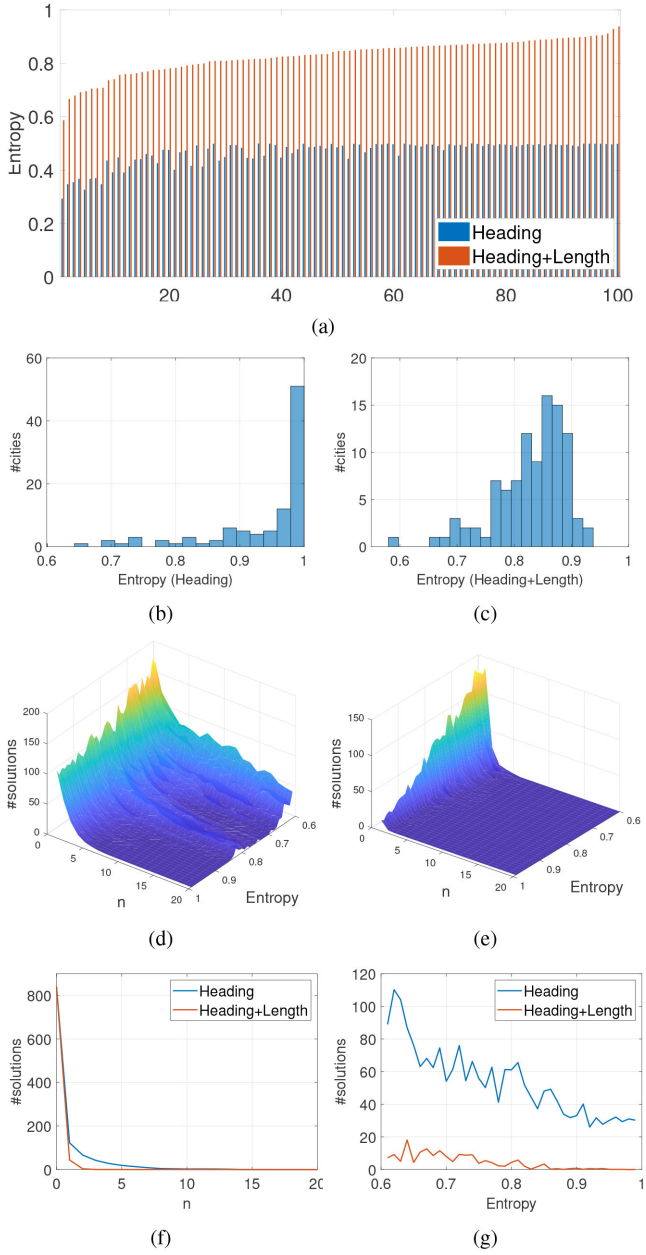


Fig. 9. (a) Entropy of 100 cities. (b) Heading entropy distribution of 100 cities. (c) Heading and length entropy distribution of 100 cites. (d) #solutions with respect to map entropy values (heading only) and n. (e) #solutions with respect to map entropy values (heading+length) and n. (f) n versus #solutions with fixed map entropy $= 0.86$. (g) Map entropy values versus #solutions with $n = 3$.

Building on the simulation in [1], we expand it from HG 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 $n = 1, \ldots, 20$ 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 $\sigma_g = 5$ m. We compute the number of solutions by averaging the results of 20 sequences for each map. The simulation result

Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

is shown in Fig. 9(e) and we adapt Fig. 9(d) from [1] for comparison.

For PLAM which uses heading only [see Fig. 9(d)], 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. 9(d) and (e) shows 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 Fig. 9(f) and (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 Corollary 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 Table I. 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 Table 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.

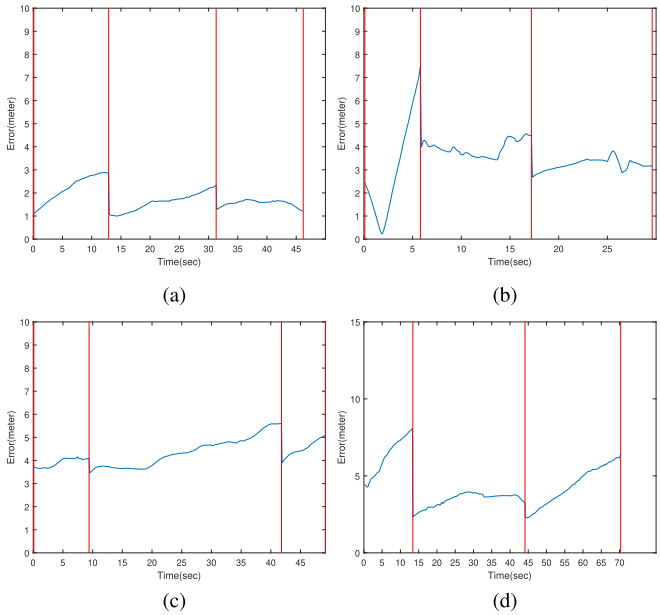


Fig. 10. LAV accuracy results using KITTI sequences on KITTI00Map and KITTI05Map. (a) KITTI00-1, (b) KITTI00-2, (c) KITTI05-1, and (d) KITTI05-2.

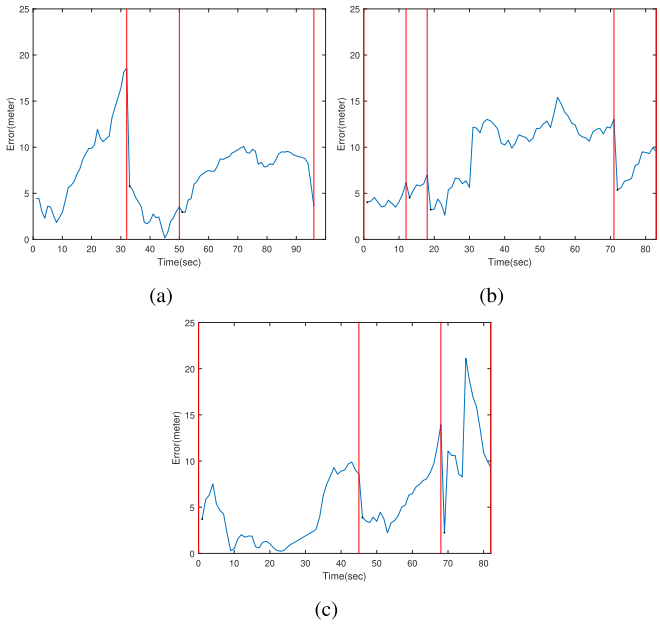


Fig. 11. LAV accuracy results using CSData on CSMap. (a) CS-1, (b) CS-2, and (c) CS-3.

B. Localization Alignment and Verification Test

GL 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.

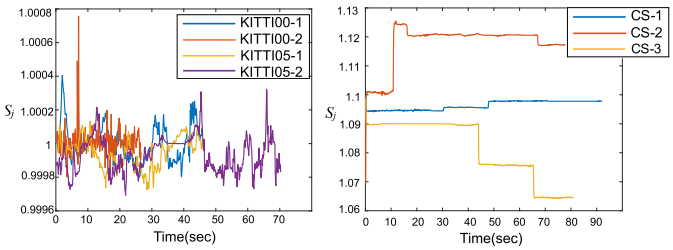
1) Ground Truth and Evaluation Metric: The ground truth in our experiments is the actual GPS trajectory. The localization error is defined as the Euclidean distance between the estimated aligned trajectory and the ground truth. The localization errors are measured in meters.

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 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 $5\tilde{}m$ 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 $10\tilde{}m$). 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 m or worse while the GPS receiver for KITTI data set is high quality GPS (model RT3000v3) with an accuracy of 1 cm.

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 vehicle velocity from the Panda OBD II dongle contains bias.

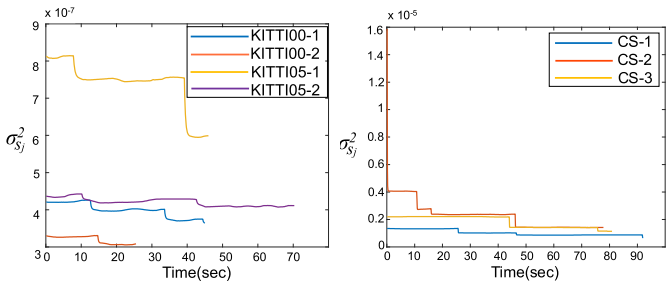
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



(a) (b)

Fig. 12. Scale and slip factor value over time in EKF (10). (a) KITTI data and (b) CSData. Note the sequences are color coded and are not of the same length in time.



(a) (b)

Fig. 13. 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.

It tends to underestimated vehicle velocity by about 10%. This may be due to incorrect parameters 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 are 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 showed the variance of s_j in Fig. 13. These results show $\sigma^2_{s_{ssf}}$ decreasing as travel length increases as in Lemma (3).

VI. Conclusion

In this article, 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 was designed to be a fallback solution when everything else failed 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 preprocessed maps into an HLG which stored all long and straight segments of road as nodes to facilitate GL process. Once the map matching was successful, our algorithm tracked vehicle movement and used 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 will actively guide the vehicle to make turns to speed up the localization process. More experiments are planed which we will work on situations of losing location and relocalization. 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

The authors 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 minimalist intermittent heading-based approach," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 586–593, Apr. 2019.

[2] M. A. Brubaker, A. Geiger, and R. Urtasun, "Map-based probabilistic visual self-localization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 4, pp. 652–665, Apr. 2016.

[3] S. Lowry *et al.*, "Visual place recognition: A survey," *IEEE Trans. Robot.*, vol. 32, no. 1, pp. 1–19, Feb. 2016.

[4] Y. Lu and D. Song, "Visual navigation using heterogeneous landmarks and unsupervised geometric constraints," *IEEE Trans. Robot.*, vol. 31, no. 3, pp. 736–749, Jun. 2015.

[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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2003, pp. 206–211.

[6] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2 D Lidar SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*,2016, pp. 1271–1278.

[7] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *Proc. IEEE Int. Conf. Robot. Autom.*,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 Trans. Intell. Transp. Syst.*, vol. 15, no. 2, pp. 507–529, Apr. 2014.

[9] Y. Cui and S. S. Ge, "Autonomous vehicle positioning with GPS in urban canyon environments," *IEEE Trans. Robot. Autom.*, vol. 19, no. 1, pp. 15–25, Feb. 2003.

[10] H. Aly, A. Basalamah, and M. Youssef, "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. Kingery, D. Wang, H. Li, and D. Song, "Encoder-camera-ground penetrating radar tri-sensor mapping for surface and subsurface transportation infrastructure inspection," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2018, pp. 1452–1457.

[12] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *Int. J. Robot. Res.*, 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2007, pp. 2845–2850.

[14] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV navigation and localization: A review," *IEEE J. Ocean. Eng.*, vol. 39, no. 1, pp. 131–149, Jan. 2014.

[15] W. Kang and Y. Han, "Smartpdr: Smartphone-based pedestrian dead reckoning for indoor localization," *IEEE Sensors J.*, vol. 15, no. 5, pp. 2906–2916, May 2015.

[16] I. Constandache, R. R. Choudhury, and I. Rhee, "CompAcc: Using mobile phone compasses and accelerometers for localization," in *Proc. 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 4868–4873.

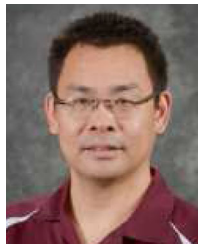
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[18] F. Gustafsson *et al.*, "Particle filters for positioning, navigation, and tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 425–437, Feb. 2002.
[19] L. Huang, B. He, and T. Zhang, "Autonomous navigation algorithm for underwater vehicles based on inertial measurement units and sonar," in *Proc. 2nd Int. Asia Conf. Inform. Control, Autom. Robot.*, 2010, pp. 311–314.
[20] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Germany: 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 *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 2787–2793.
[22] P. Ruchti, B. Steder, M. Ruhnke, and W. Burgard, "Localization on openstreetmap data using a 3 D laser scanner," in *Proc. IEEE Int. Conf. Robot. Autom.*, 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 Robot. Autom. Lett.*, vol. 2, no. 2, pp. 789–795, Apr. 2017.
[24] Y. Jin, and Z. Xiang, "Robust localization via turning point filtering with road map," in *Proc. IEEE Intell. Veh. Symp.*, 2016, pp. 992–997.
[25] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.
[26] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org," 2017. [Online]. Available: https://www.openstreetmap.org
[27] Google Maps Contributors, 2017. [Online]. Available: https://www.google.com/maps/
[28] M. Quddus and S. Washington, "Shortest path and vehicle trajectory aided map-matching for low frequency GPS data," *Transp. Res. Part C: Emerg. Technol.*, 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 *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.
[30] J. Wahlström, I. Skog, J. G. P. Rodrigues, P. Händel, and A. Aguiar, "Map-aided dead-reckoning using only measurements of speed," *IEEE Trans. Intell. Veh.*, vol. 1, no. 3, pp. 244–253, Sep. 2016.
[31] B. Yu *et al.*, "A hybrid dead reckoning error correction scheme based on extended Kalman filter and map matching for vehicle self-localization," *J. Intell. Transp. Syst.*, vol. 23, no. 1, pp. 84–98, 2019.
[32] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[33] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation With Applications to Tracking and Navigation: Theory Algorithms and Software*. Hoboken, NJ, USA: Wiley, 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 Trans. Robot.*, vol. 25, no. 5, pp. 1087–1097, Oct. 2009.
[35] H.-M. Cheng and D. Song, "Localization in inconsistent Wifi environments," in *Proc. Int. Symp. Robot. Res.*, 2017, pp. 661–678.
[36] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man, Cybern.*, vol. 9, no. 1, pp. 62–66, Jan. 1979.
[37] R. A. Kronmal and A. V. Peterson, Jr., "On the alias method for generating random variables from a discrete distribution," *Amer. Statis.*, vol. 33, no. 4, pp. 214–218, 1979.
[38] G. Boeing, "Urban Spatial Order: Street Network Orientation, Configuration, and Entropy," *Appl. Network Sci.*, vol. 4, no. 1, 2019, Art. no. 67.
[39] N. Mohajeri and A. Gudmundsson, "The evolution and complexity of urban street networks," *Geographical Anal.*, vol. 46, no. 4, pp. 345–367, 2014.
[40] J. Borenstein and L. Feng, "Correction of systematic odometry errors in mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. Human Robot Interact. Cooperative Robots*, 1995, pp. 569–574.
[41] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The Kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

Hsin-Min Cheng received the B.S. and M.S. degrees in electronic engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2010 and 2012, respectively. She is currently working toward the Ph.D. degree with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA.

Her research interests include robot perception, sensor fusion, and localization.



Dezhen Song (Senior Member, IEEE) received the Ph.D. degree in industrial engineering and operations research from the University of California, Berkeley, CA, USA, in 2004.

He is a Professor with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA. His research interests include networked robots, multimodal perception, computer vision, and stochastic modeling.

Dr. Song was the recipient of the Kayamori Best Paper Award of the 2005 IEEE International Conference on Robotics and Automation (with J. Yi and S. Ding). He received NSF Faculty Early Career Development (CAREER) Award in 2007. From 2008 to 2012, he was an Associate Editor of IEEE TRANSACTIONS ON ROBOTICS. From 2010 to 2014, he was an Associate Editor of IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING. From 2017 to 2020, he was a Senior Editor for *IEEE Robotics and Automation Letters (RA-L)*.

Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on May 16,2021 at 09:30:25 UTC from IEEE Xplore. Restrictions apply.