Onboard Sensors-Based Self-Localization for Autonomous Vehicle With Hierarchical Map

Chao Xia, Yanqing Shen, Yuedong Yang, Xiaodong Deng, Shitao Chen, *Member, IEEE*, Jingmin Xin, *Senior Member, IEEE*, and Nanning Zheng, *Fellow, IEEE*

***Abstract*—Localization is a fundamental and crucial module for autonomous vehicles. Most of the existing localization methodologies, such as signal-dependent methods (RTK-GPS and Bluetooth), simultaneous localization and mapping (SLAM), and map-based methods, have been utilized in outdoor autonomous driving vehicles and indoor robot positioning. However, they suffer from severe limitations, such as signal-blocked scenes of GPS, computing resource occupation explosion in large-scale scenarios, intolerable time delay, and registration divergence of SLAM/map-based methods. In this article, a self-localization framework, without relying on GPS or any other wireless signals, is proposed. We demonstrate that the proposed homogeneous normal distribution transform algorithm and two-way information interaction mechanism could achieve centimeter-level localization accuracy, which reaches the requirement of autonomous vehicle localization for instantaneity and robustness. In addition, benefitting from hardware and software co-design, the proposed localization approach is extremely light-weighted enough to be operated on an embedded computing system, which is different from other LiDAR localization methods relying on high-performance CPU/GPU. Experiments on a public dataset (Baidu Apollo SouthBay dataset) and real-world verified the effectiveness and advantages of our approach compared with other similar algorithms.**

***Index Terms*—Autonomous vehicle localization, homogeneous registration method, normal distribution transform (NDT)-EKF tightly coupled algorithm, software–hardware co-design.**

I. Introduction

AUTONOMOUS driving technology, as one of the best verification solutions for artificial intelligence [1]–[3] and cognitive science [4], [5], has made great progress in recent decades, where localization is a prerequisite for safe driving in large-scale urban scenes. As is well known, autonomous driving technology consists of four main parts, including controller, path planning, localization, and perception [6]. The controller corrects the error between the vehicle position and the intended trajectory [7]–[12], where real-time localization result serves as the controller's feedback. Algorithms are designed to overcome the delay and error of pose estimation, such as fuzzy logic and fuzzy controller [13]–[15] because localization delay and error are inevitable.

For a long time, the real-time kinematic (RTK) technique has provided localization services by centimeter accuracy for autonomous driving vehicles. However, there are too many GPS-denied environments, for instance, skyscrapers, tree-lined roads, and underground garages, which significantly affect the accuracy and robustness of autonomous vehicle localization. The same problem also arises in other signal-dependent methods, such as Bluetooth, ZigBee, WLAN, and UWB. And these infrastructure-dependent methods are disadvantageous in large-scale outdoor scenes for autonomous vehicles.

Simultaneous localization and mapping (SLAM) [16]–[20] is another type of method, which realizes map construction and interframe pose estimation through camera/LiDAR scan-matching. But due to the low frequency, SLAM is typically used for map building rather than localization, particularly for outdoor scenes. Compared with SLAM, map-based methods estimate vehicle pose by matching the real-time scan and prebuilt high-definition (HD) map; and these methods are also called self-localization. However, both SLAM and map-based methods will inevitably result in an explosion of computing and storage resources as the stage scale increases. In particular, some problems such as storing and loading large-scale HD maps, real-time and robust scan–map registration in sparse or monotonic scenes, hinder the promotion of self-localization methods in real urban traffic environments.

Generally speaking, the localization output frequency should reach 100 Hz or higher, along with centimeter-level accuracy. Otherwise, the controller cannot be able to drive smoothly and safely, especially at high speeds. Considering the problems of existing localization methods and the performance metrics requirements of autonomous vehicle localization, a self-localization framework based on software and hardware co-design is proposed in this study. A comparison between our self-localization framework and other widely used localization methods is shown in Table I. Through the improvement of map construction, scan–map registration, and tightly coupled HNDT-EKF algorithm, the proposed localization framework can solve the divergence problem of the registration algorithm efficiently, especially in dynamic and sparse scenes, namely, the corner case. To improve the energy efficiency ratio of the algorithm, we have followed the idea of software and hardware

Manuscript received July 15, 2021; revised November 11, 2021 and January 12, 2022; accepted February 26, 2022. This work was supported by the National Natural Science Foundation of China under Grant 62088102 and Grant 61773312. This article was recommended by Associate Editor R.-E. Precup. *(Corresponding author: Nanning Zheng.)*

The authors are with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: xc06210417@stu.xjtu.edu.cn; qing1159364090@stu.xjtu.edu.cn; invincibleyang@stu.xjtu.edu.cn; dinger@stu.xjtu.edu.cn; chenshitao@mail.xjtu.edu.cn; jxin@mail.xjtu.edu.cn; nnzheng@mail.xjtu.edu.cn).

Digital Object Identifier 10.1109/TCYB.2022.3155724

TABLE I
OVERVIEW OF CURRENT AUTONOMOUS VEHICLE LOCALIZATION METHODS AND THEIR PROBLEMS. OUR PROPOSALS AND THEIR ADVANTAGES ARE LISTED IN THE REAR

Method / Property	RTK-GPS	Wireless	SLAM	Map-based	Proposals	Advantage
External dependency	√	√			Self-localization	Only relying on onboard sensors
Accuracy	High	Low	Low	High	Homogeneous method	Higher accuracy and faster
Computational cost	Low	Medium	High	High	Software and hardware co-design	Higher efficiency and lower power consumption
Storage occupation	Low	Low	High	High	GeoHash-based map division and encoding	Lower occupation and faster indexing
Stability	Medium	Medium	Low	Medium	NDT-EKF tightly-coupled	More stable and robust

co-design [21] in our previous work, where the proposed self-localization framework is deployed on an advanced RISC machine (ARM) with a field-programmable gate array (FPGA) computing platform.

The contributions of this article are summarized as follows.

1) *Hierarchical Map Building:* The GeoHash-based map partition and encoding method is utilized to achieve large-scale map division, compression, and fast submap indexing. Experiments have shown that our method reduces the storage space occupation by 99.96% and computing time consumption by 99.70% compared with the KD-TREE-based nearest neighbor search (NNS) on point cloud.
2) *Homogeneous Method:* To improve the accuracy and robustness of the registration algorithm, a homogeneous method is applied on both normal distribution transform (NDT) and iterative closest point (ICP). In addition, through highly parallel and pipeline design on FPGA, the latency can be reduced to extremely low.
3) *Multisensor Fusion:* An NDT-EKF tightly coupled algorithm is proposed to improve the robustness of the localization framework. In the two-way information interaction between NDT and EKF, the corresponding covariance is transmitted along with pose. The uncertainty of the LiDAR localization result is taken into consideration in EKF, while covariance of state estimation in EKF is added to the loss function of the LiDAR localization algorithm.

II. RELATED WORK

RTK, a carrier-phase-based differential global navigation satellite system (GNSS) technique [22], can provide centimeter-level localization accuracy in outdoor scenes. By fusing RTK-GPS with an inertial measurement unit (IMU), an integrated navigation system (INS) can promote the frequency and smoothness of localization results, which is one of the most widely used localization methods [23], [24]. Besides GPS, lots of wireless technologies, including Bluetooth, ZigBee, WLAN, and UWB [25]–[28], are employed to pose estimation. To solve the problem that statistical parameters of noise are not available, Yang *et al.* [29] proposed distributed set-membership filtering and verified the effectiveness in indoor environments.

In recent years, map-based LiDAR localization [30]–[33] has been developed, which is called upon to solve the problem of localization drafting caused by signal occlusion. Point cloud registration is the key LiDAR localization, including point-to-point (P2P) [34]–[37], point-to-distribution (P2D) [38], [39], model-based [40], [41], and learning-based methods [42], [43]. However, most of them suffer from high delay for large-scale scan–map matching. The more serious problem is that the registration divergence cannot be effectively solved, which is dangerous for LiDAR localization. Researchers have been attempting to improve the accuracy, robustness [44]–[46], and speed up registration methods by CPU or GPU [47], [48]. However, their requirements on the computing power are too high, and it is difficult for the onboard computing platform to satisfy. In addition, LOAM [16], proposed by Zhang and Singh, achieves point cloud registration by extracting features, such as points, lines, and surfaces, which has been successfully applied to LiDAR SLAM, but it can be only used for scan–scan matching but not scan–map matching. Wolcott and Eustice [49], [50] proposed a fast multiresolution scan matcher using Gaussian mixture maps for vehicle localization. Lu *et al.* proposed a learning-based LiDAR localization system that achieves centimeter-level localization accuracy by using various deep neural network structures, which is dependent on high-performance GPU.

Obviously, it seems that high precision and low power cannot be satisfied simultaneously for a LiDAR localization method. Therefore, improving the accuracy and robustness of localization result, and reducing the computing power simultaneously is the main problem to be solved in this study. Considering the power limitation of onboard computing platform, the proposed localization framework can be implemented on an embedded system (ARM + FPGA), whose power is only 10 W. In addition, limited by the frequency of the LiDAR itself (5–20 Hz), to achieve high-frequency localization result, the LiDAR localization output shall be passed to the sensor fusion algorithm [28], [51]–[55] as an observation. In fact, this one-way information transmission is postprocessing for LiDAR localization [56], [57], but the performance of LiDAR localization itself will not be improved. According to our previous work [58], this article puts forth the NDT-EKF tightly coupled algorithm dependent on two-way information interaction, which improves the accuracy and robustness of both LiDAR localization and multisensor fusion results.

III. Problem Statement

The self-localization refers to a localization system that uses onboard sensors, such as LiDAR, camera, IMU, wheel-speed-odometer, etc., to obtain accurate vehicle poses through real-time environment perception and navigation map matching. The key to the self-localization system is how to achieve robust and accurate scan–map matching. However, it suffers from data-intensive and divergence problems. On the one hand, data-intensive computing prompts the registration algorithm acceleration by high-performance CPU or GPU, whose power is already too high for a robot or vehicle. On the other hand, registration divergence directly leads to localization drift, which is dangerous for autonomous driving vehicles.

Owing to the dynamics and sparseness of the real traffic scenes, registration divergence seems inevitable for all geometric registration methods. To address these contradictions, a more robust registration method combined with acceleration on FPGA will be discussed in this article. In general, a self-localization algorithm flow is illustrated in Algorithm 1. Our improvements will be concentrated on function “REG” and “MSF,” which will be discussed detailedly in Sections IV-B and IV-C. In addition, software and hardware co-design is another essential part, which ensures that the proposed self-localization framework can run on a low-power hardware platform.

Algorithm 1 Self-Localization Algorithm

```
Input: $\mathcal{D}$: Raw data from sensors; $\mathcal{M}$: Pre-built hierarchical
    map; $\mathbf{X}_{init}$: Initial pose;
Output: $\mathbf{X}$: Real-time pose estimation
 1: while Sensors are all available do
 2:     $\mathbf{X}$ = SLA($\mathcal{D}$, $\mathcal{M}$, $\mathbf{X}_{init}$)
 3: end while
 4: function SLA($\mathcal{D}$, $\mathcal{M}$, $\mathbf{X}_{init}$)
 5:     if $\mathcal{M}$ and $\mathbf{X}_{init}$ then
 6:         Search sub-map $\mathcal{M}_{sub}$ from $\mathcal{M}$ according to $\mathbf{X}_{init}$
 7:     end if
 8:     Raw data pre-processing: $\mathcal{D}_p \leftarrow \mathcal{D}$
 9:     if $\mathcal{M}_{sub}$ and $\mathcal{D}_p$ then
10:         $\mathcal{L}_{lidar}$ = REG($\mathcal{D}_p$, $\mathcal{M}_{sub}$)
11:         $\mathcal{L}_{msf}$ = MSF($\mathcal{D}_p$, $\mathcal{L}_{lidar}$)
12:         $\mathbf{X} \leftarrow \mathcal{L}_{msf}$
13:     end if
14:     return $\mathbf{X}$
15: end function
16: function REG($\mathcal{D}_p$, $\mathcal{M}_{sub}$)
17:     Real-time LiDAR scan and sub-map matching;
18:     return $\mathcal{L}_{lidar}$
19: end function
20: function MSF($\mathcal{D}_p$, $\mathcal{L}_{lidar}$)
21:     Predict current states based on the vehicle kinematics
    model and last states;
22:     Correct the predicted states by $\mathcal{L}_{lidar}$;
23:     return $\mathcal{L}_{msf}$
24: end function
```

IV. Self-Localization Framework

Three submodules constitute the self-localization framework, including submap update, LiDAR localization, and sensor fusion. Thanks to the software and hardware co-design, the LiDAR localization submodule can run at the same frequency as the LiDAR raw data update (e.g., 10 Hz). It could reduce latency in the LiDAR localization submodule through parallel computing. Then, the LiDAR localization result and raw data from the IMU and wheel speed are used for updating and propagation respectively in the multisensor fusion submodule. The estimated pose by multisensor fusion is the final localization consequence, which is then delivered to subsequent modules, such as path planning and vehicle controller. In the meanwhile, it serves as the initial pose for the scan–map registration at the next frame. The flow diagram is shown in Fig. 1.

A. Hierarchical Map Building and Local Map Update

Saving a large-scale global map and finding a corresponding submap according to the vehicle current position dynamically and efficiently is a crucial ability. In general, a global map can be divided into voxels, and the normal distribution (ND) parameter can be calculated for valid voxels (point size in a voxel larger than a threshold). Compared with saving point cloud directly, the memory occupation of this method is significantly reduced. Analogously, submap quickly indexing from numerous unordered voxels is another problem. According to our previous work [59], to realize fast submap indexing, GeoHash, a public domain geocode system, has been used to encode all valid voxels geographic location into a binary string. In this study, two different resolution domains (block and voxel) have been defined; the block is the larger domain, and a valid block must contain enough valid voxels. The multiresolution map is also called a hierarchical navigation map.

In contrast to the original GeoHash, the encoding method used in this study is a binary space partitioning method, which divides the global map into 2-D blocks ($X-Y$, 24 m $\times$ 24 m) and each of them has a short code as its index that represents its geographic location. After that, every block is subdivided into 3-D voxels ($X-Y-Z$, 1.5 m$\times$1.5 m$\times$1.5 m) and each of them has a long code that also represents geographic location of a voxel. Blocks contain all the voxels within its “$x-y$” area, regardless of the height, or they can be regarded as cubes with an infinite height. The global map division, encoding, and data structure are shown in Fig. 2. The three figures on the left [Fig. 2(a)–(c)] show partial zooming and division of the global map from top to bottom. Blocks with the blue mask are invalid,

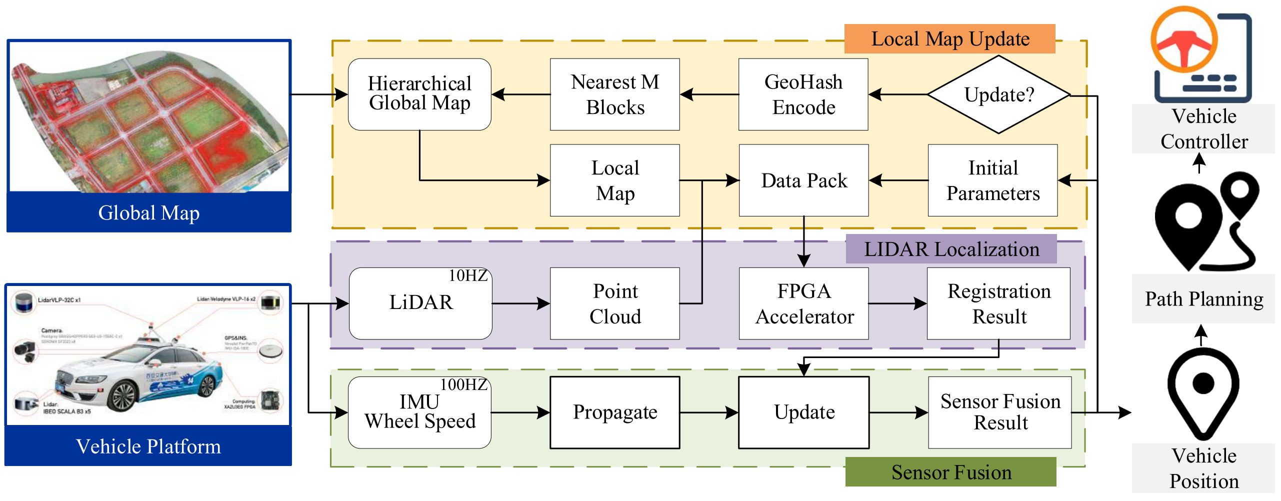


Fig. 1. Flow diagram of our self-localization framework. It contains three main submodules, submap update (yellow), LiDAR localization (purple), and multisensor fusion (green), which are running at different frequencies.

and the remaining blocks are valid. The right middle [Fig. 2(e)] and right bottom [Fig. 2(f)] figures are valid blocks and voxels data structures, respectively. The right top [Fig. 2(d)] is an example of a 4×4 block encoding method and the symbol "$*$" in the front of each binary string represents an identical prefix. In fact, the hierarchical map, blocks, and voxels are all 3-D structures, and they are all displayed in 2-D for visualization purpose.

In the proposed self-localization framework, the submap (as a target point cloud in the registration) needs to be updated along with the autonomous vehicle driving (e.g., update every 10 m) to ensure that the scan and map represent the same area. When the submap update condition is triggered, the vehicle current position will be encoded into a short code, which is the same as the block encoding method. From the discussion above, this code contains location information of the vehicle, then the nearest M blocks can be found quickly. In addition, these blocks contain the indices of those valid voxels (long code), which are delivered to the FPGA module through direct memory access (DMA). Finally, on FPGA, a double hash function is utilized to realize a mapping between the memory address and voxel index, which makes it possible for the corresponding voxel address to be found quickly and accurately. As a result of these processes, the statistics and position information of corresponding voxels contained in a submap can be found for the subsequent registration algorithm.

Here, the GeoHash-based map division and location encoding method can not only achieve map data compression but also build a one-to-one correspondence between location and memory address. Therefore, the double-hash function can be used for fast submap searching, which makes it possible to store a city-scale global navigation map and dynamically load a suitable submap with minimal consumption of time and computing resources.

B. LiDAR Localization

It is difficult for existing registration algorithms to meet the requirements of real-time localization in terms of robustness and accuracy. The most important reason for the failure of the registration algorithm is the inhomogeneous data distribution of the scene described by the point cloud. Algorithms such as generalized ICP (GICP) [44] and NDT [38] take into account the distribution information of local adjacent points for registration. But to the best of our knowledge, no geometric algorithm has considered the impact of global point

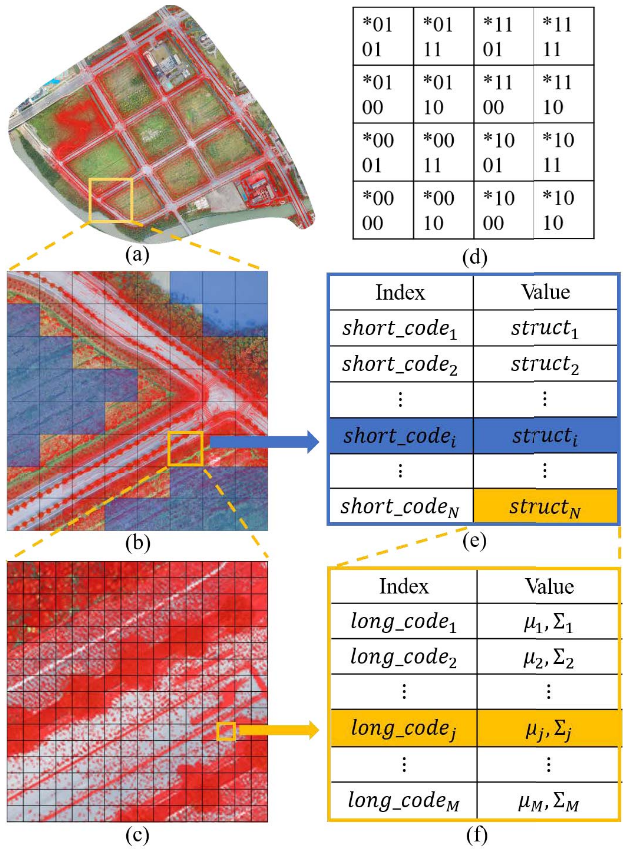


Fig. 2. Hierarchical map construction diagram. Two different resolutions of map division exist in it, including blocks and voxels. (a) Global map. (b) Valid and invalid (with blue mask) blocks. (c) Valid block zooming up. (d) Example of encoding. (e) Block data structure. (f) Voxel data structure.

cloud distribution in a scene, which is a decisive factor for the performance of registration algorithms. Therefore, a homogeneous method that modifies the gradient and adjusts the convergence direction in each iteration according to global point cloud distribution information has been proposed, which can be applied to both the original ICP and NDT.

1) Homogeneous ICP: Assuming the transformation matrix $\mathbf{X} = \begin{bmatrix} \mathbf{Rot} & \boldsymbol{t} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \in \mathbb{SE}(3) | \mathbf{Rot} \in \mathbb{SO}(3), \boldsymbol{t} \in \mathbb{R}^{3\times 1}$ to be estimated aligns source $\mathcal{P} = \{\boldsymbol{p}_j | j = 1, 2, \ldots, N_p\}$ and target $\mathcal{Q} = \{\boldsymbol{q}_j | j = 1, 2, \ldots, N_q\}$ point cloud. And the correspondences between $\mathcal{P}$ and $\mathcal{Q}$ given by NNS are defined as $\mathcal{Q}' = \mathrm{T}(\mathbf{X}, \mathcal{P}')$, where $\mathcal{P}' = \{\boldsymbol{p}_i | i = 1, 2, \ldots, N_k\}$ and $\mathcal{Q}' = \{\boldsymbol{q}_i | i = 1, 2, \ldots, N_k\}$ are subsets of $\mathcal{P}$ and $\mathcal{Q}$, respectively. $\mathrm{T}(\mathbf{X}, \mathcal{A})$ indicate a coordinate transformation processing on a point cloud $\mathcal{A}$ by a transformation matrix $\mathbf{X}$. The objective function can be written as

$$\hat{\mathbf{X}}_k = \underset{\mathbf{Rot}_k, \boldsymbol{t}_k}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left\| \boldsymbol{p}_i - \left(\mathbf{Rot}_k \cdot \boldsymbol{q}_i + \boldsymbol{t}_k \right) \right\|^2. \quad (1)$$

This optimization problem shown in (1) has a closed-form solution, which can be solved by a two-step method to obtain the rotation matrix and the translation vector separately. In the first step, the rotation matrix $\hat{\mathbf{Rot}}$ can be solved by SVD decomposition. Then in second step, the translation vector $\hat{\boldsymbol{t}}$ can be easily solved: $\hat{\boldsymbol{t}} = \boldsymbol{p} - \hat{\mathbf{Rot}} \cdot \boldsymbol{q}$, where $\boldsymbol{p}$ and $\boldsymbol{q}$ are the centroids of $\mathcal{P}'$ and $\mathcal{Q}'$, respectively. Apparently, the translation vector is determined by the centroids of correspondence given by NNS. However, the problem is that there are too many mismatching point pairs between real-time LiDAR scan and submap, which is therefore difficult to converge to the global optimal solution, especially in sparse scenes. Hence, a homogeneous method is proposed to eliminate the impact of the mismatching point pairs by considering the global point cloud distribution in a scene.

First, each point $\boldsymbol{p}_j$ in source and its K nearest neighbors (KNNs) $\mathcal{Q}_j = \{\boldsymbol{q}_k | \boldsymbol{q}_k \in \mathcal{Q}, k = 1, 2, \ldots, K\}$ in target can be defined as a new set $\mathcal{G}_j = \{\boldsymbol{p}_j, \mathcal{Q}_j\}$; then, the mean $\boldsymbol{\mu}_j \in \mathbb{R}^{3\times 1}$ and covariance matrix $\mathbf{C}_j \in \mathbb{R}^{3\times 3}$ of Q_j can be calculated as

$$\boldsymbol{\mu}_j = \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{q}_k \quad (2)$$

$$\mathbf{C}_j = \frac{1}{K} \sum_{k=1}^{K} \left(\boldsymbol{q}_k - \boldsymbol{\mu}_j \right) \left(\boldsymbol{q}_k - \boldsymbol{\mu}_j \right)^{\mathrm{T}}. \quad (3)$$

Then, translation gradient $\Delta \boldsymbol{t}_j$ generated by each $\mathcal{G}_j$ can be calculated as

$$\Delta \boldsymbol{t}_j = \mathbf{C}_j^{-0.5} \Delta \boldsymbol{p}_j \quad (4)$$

where $\Delta \boldsymbol{p}_j = \mathrm{T}(\mathbf{X}, \boldsymbol{p}_j) - \boldsymbol{\mu}_j$. Equation (4) can be illustrated as a gradient vector generated by a point-to-distribution projection, and the matrix $\mathbf{C}_j^{-0.5}$ is used for gradient vector normalization.

Finally, to ensure the homogeneity of the final translation gradient vector in all directions after normalization, the total gradient of translation $\Delta \boldsymbol{t}_{\mathrm{sum}}$ in the kth iteration is modified by the sum of $\Delta \boldsymbol{t}_j$

$$\Delta \boldsymbol{t} = \mathbf{W}^{-1} \sum_{j=1}^{N_p} \Delta \boldsymbol{t}_j \quad (5)$$

where $\mathbf{W} = \sum_{j=1}^{N_p} \mathbf{C}_j^{-0.5}$. $\mathbf{W}^{-1} \sum_{j=1}^{N_p} \mathbf{C}_j^{-0.5} \Delta \boldsymbol{p}_j$ in (5) is the homogeneous matrix, which can effectively reduce oscillations in the iteration process. The optimal translation in the kth iteration is

$$\hat{\boldsymbol{t}} = \boldsymbol{p} - \hat{\mathbf{Rot}} \cdot \boldsymbol{q} + \Delta \boldsymbol{t}. \quad (6)$$

Algorithm 2 HICP Algorithm

```
Input: $\mathcal{A} = \{\boldsymbol{a}_j | j = 1, 2, \cdots, N\}$: Source point cloud;
    $\mathcal{B} = \{\boldsymbol{b}_j | j = 1, 2, \cdots, N\}$: Target point cloud; $\mathbf{X}_{init}$: Initial
    transformation matrix between source and target;
Output: $\mathbf{X} | (\mathbf{Rot}, \boldsymbol{t})$: Transformation matrix
1: $\mathbf{X} \leftarrow \mathbf{X}_{init}$
2: $\mathrm{T}(\mathcal{A}, \mathbf{X})$
3: while not converged do
4:     for $i = 1 : 1 : N$ do
5:         $\boldsymbol{m}_i \leftarrow \mathrm{NNS}(\mathcal{B})$
6:     end for
7:     First step: $\hat{\mathbf{Rot}} \leftarrow \arg\min \sum_i || \mathrm{T}(\mathbf{X}, \boldsymbol{a}_i) - \boldsymbol{m}_i ||^2$
8:     for $i = 1 : 1 : N$ do
9:         Find nearest K points: $\mathcal{C}_i \leftarrow \mathrm{KNNS}(\boldsymbol{a}_i, \mathcal{B})$
10:        $\boldsymbol{\mu}_i \leftarrow \mathrm{MEAN}(\mathcal{C}_i)$, $\mathbf{C}_i \leftarrow \mathrm{COV}(\mathcal{C}_i)$
11:        $\Delta \boldsymbol{p}_i = \boldsymbol{a}_i - \boldsymbol{\mu}_i$
12:    end for
13:    Homogeneous matrix: $\mathbf{W} = \sum_{j=1}^{N} \mathbf{C}_j^{-0.5}$
14:    Total gradient offset: $\Delta \boldsymbol{t} = \mathbf{W}^{-1} \sum_{j=1}^{N} \mathbf{C}_j^{-0.5} \Delta \boldsymbol{p}_j$
15:    Second step: $\hat{\boldsymbol{t}} = \boldsymbol{p} - \hat{\mathbf{Rot}} \cdot \boldsymbol{q} + \Delta \boldsymbol{t}$
16:    $\mathbf{X} \leftarrow (\mathbf{Rot}, \boldsymbol{t})$
17:    then $\mathrm{T}(\mathcal{A}, \mathbf{X})$
18: end while
```

The complete solving process of homogeneous ICP (HICP) algorithm is listed as Algorithm 2. Through the above improvement, the translation gradient produced by each set of points to the centroid of distribution $\Delta \boldsymbol{p}_j$ is corrected by the local point cloud distribution information $\mathbf{C}_j^{-0.5}$, and to guarantee the homogeneity of the summary of all local gradients, the final translation gradient should be corrected by $\mathbf{W}^{-1}$.

2) Homogeneous NDT: HICP has been proposed above, but the problems are also obvious. For example, HICP is still a two-step solution, and only translation gradient has been changed in the HICP solution process while rotation is still consistent with the original ICP. Moreover, it needs to perform numerous times in KNNS and SVD decomposition for every covariance matrix in each iteration, which can achieve quick solutions on high-performance CPU, but cannot be operated on a low-power computing platform such as FPGA. However, it is enlightening that through the joint action of local and global point cloud distribution information, the homogeneous method can speed up the registration algorithm convergence and obtain more accurate results. It is well known that NDT [38] is another P2D registration method. In contrast to P2P NNS, NDT only needs to find the corresponding voxel (the number of voxels is much smaller than the point) $\mathcal{V}_j|(\boldsymbol{\mu}_j, \mathbf{C}_j)) \in \mathcal{V} = \{\mathcal{V}_k|(\boldsymbol{\mu}_k, \mathbf{C}_k), k = 1, 2, \ldots, N_v\}$ for each point in the source $\boldsymbol{p}_j$ according to the initial pose estimation. The original NDT loss function is

$$\mathrm{s}(\mathbf{X}) = -\sum_{j=1}^{n} \tilde{\mathrm{p}}\big(\mathrm{T}(\mathbf{X}, \boldsymbol{p}_j)\big) \quad (7)$$

$$\tilde{\mathrm{p}}(\boldsymbol{p}_j) = -d_1 \exp\left(-\frac{d_2}{2} \Delta \boldsymbol{p}_j^{\mathrm{T}} \mathbf{C}_j^{-1} \Delta \boldsymbol{p}_j\right) \quad (8)$$

where d_1 and d_2 are constants, which can be estimated by parameter fitting easily. The exponent part of the loss function is the Mahalanobis distance between point $\boldsymbol{p}_j$ and its corresponding ND $\mathcal{V}_j$. It simply sums the "distance" generated by each P2D projection, but does not take the global data distribution information into account. As previously shown, local P2D without a global weighted average cannot solve the divergence problem owing to the inhomogeneous data distribution of the global scene. Therefore, the homogeneous NDT (HNDT) loss function can be reconstructed as follows:

$$\begin{aligned}\mathrm{J}(\hat{\mathbf{X}}) &= \sum_{j=1}^{N_p} d_1 \exp\left\{-\frac{d_2}{2}\left(\mathbf{W}^{-1}\mathbf{C}_j^{-0.5}\Delta \boldsymbol{p}_j\right)\right)^{\mathrm{T}}\left(\mathbf{W}^{-1}\mathbf{C}_j^{-0.5}\Delta \boldsymbol{p}_j\right)\right\} \\ &= \sum_{j=1}^{N_p} d_1 \exp\left\{-\frac{d_2}{2}(\Delta \boldsymbol{p}_j)^{\mathrm{T}}\left(\mathbf{C}_j^{0.5}\mathbf{W}^2\mathbf{C}_j^{0.5}\right)^{-1}(\Delta \boldsymbol{p}_j)\right\} \quad (9)\end{aligned}$$

where $\mathbf{W} = \sum_{j=1}^{N_p} \mathbf{C}_j^{-0.5}$. According to Newton–Raphson (NR), at the kth step of NR, the current estimate of the transformation parameter $\mathbf{X}^k$ is updated as

$$\mathbf{X}^{k+1} = \mathbf{X}^k + [-\mathbf{H}]^{-1}\boldsymbol{g}. \quad (10)$$

Substitute (10) into HNDT loss function (9)

$$\sum_j \mathbf{H}_j^k \Delta \mathbf{X}^k = -\sum_j \boldsymbol{g}_j^k \quad (11)$$

where $\mathbf{H}_j^k$ and $\boldsymbol{g}_j^k$ are the Hessian matrix and Jacobian matrix generated by point $\boldsymbol{p}_j$ and its corresponding voxel $\mathcal{V}_j$ in (9) for the kth iteration, respectively.

From (8) and (9), we can see that the loss functions of HNDT and original NDT are consistent in mathematical form. The only difference is that the covariance matrix is weighted using distribution information in a submap instead of a single voxel; thus, the solution process and the execution efficiency of the algorithm will not be affected. In fact, HNDT and HICP are consistent in terms of their algebraic forms in homogeneous processing as well. Considering all the covariance matrices in a submap, $\mathbf{W}^{-1}$ could impact the Hessian and Jacobian matrix calculation in each iteration, which means that the gradient $\sum \Delta \boldsymbol{p}_j^k$ generated by sets of points and distributions will be more homogeneous in all directions.

Fig. 3 shows the convergence process comparison between the original ICP, NDT, and our HICP, HNDT with the same parameters (convergence condition, maximum number of iterations, and initial pose) and scene [real-time LiDAR scan (source) and submap (target)]. The initial position (x–y–z, xyz) and orientation (yaw–pitch–roll, ypr) are zero, and the ground truth is xyz = [0.5, 0.5, 0] (m) and ypr = [0.1, 0, 0] (rad). As we can see, compared with the original ICP and NDT, the gradient direction generated by the homogeneous method was rectified in each iteration, and the final registration result was more accurate. In addition, the accuracy of the final registration result (position and orientation) from HNDT and HICP is higher than the original methods, which proves the effectiveness of the homogeneous method. The iteration trajectory of HNDT is smoother than that of NDT owing to the homogeneous matrix added to the loss function.

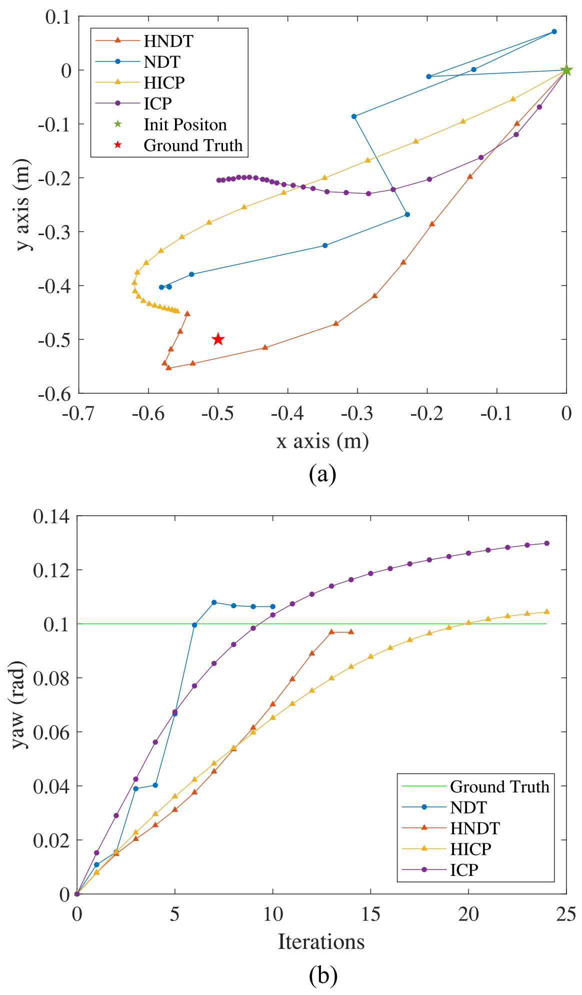


Fig. 3. Convergence process comparison of ICP, NDT, and our HICP and HNDT. (a) Position. (b) Direction.

C. HNDT-EKF Tightly Coupled Algorithm

Through the improvements of the above two sections (Sections IV-A and IV-B), a more accurate and robust LiDAR localization result can be obtained. However, owing to the low frequency and the measurement error of the LiDAR itself, it is not sufficient to satisfy the requirement of autonomous vehicle pose estimation using LiDAR localization alone. Therefore, multisensor fusion was conducted to provide smoother and higher-frequency localization results. In this study, the EKF is used to fuse the LiDAR localization result with the wheel speed and IMU, in which the IMU and wheel speed are used for state prediction, and the LiDAR localization result is used for state update.

1) Pose and Its Confidence Estimation From Scan–Map Registration: In the processing observations part, $\mathbf{R}$, the covariance matrix of observations in EKF, is used to represent the uncertainty of observation, and accordingly influences the subsequent weighted calculation. It can be assumed that when there is a mismatch observation with its uncertainty substituted into the weighted calculation, the accuracy of the fusion results will decrease, especially when the inaccurate observation information is given a low uncertainty. In practical applications, $\mathbf{R}$ is generally set to a constant matrix. In view of corner cases in real-world driving scenarios, such as highways, there may be a deviation between the ground truth and the LiDAR localization, resulting in fluctuations in the credibility of the result. Therefore, the covariance matrix of the registration result should also be input into the EKF synchronously to dynamically affect the weight of the observations in the fusion calculation. According to the NR method, when the convergence condition of optimization iteration is reached, namely, $\Delta\mathbf{X}^{k+1} \approx \Delta\mathbf{X}^{k}$, the inverse of the Hessian provides an estimated covariance matrix [57]

$$\mathbf{R} = \lambda\left(\frac{\partial^2 \mathrm{J}(\mathbf{X})}{\partial \mathbf{X}^2}\bigg|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k}\right)^{-1} = \lambda\sum_j\left(\mathbf{H}_j^{\mathcal{L}}\right)^{-1} \tag{12}$$

where λ is a proportional parameter and $\mathbf{H}_j^{\mathcal{L}}$ is the Hessian matrix generated by a set of points $\boldsymbol{p}_j$ and its corresponding voxel $\mathcal{V}_j$ in the last iteration.

2) Initial Pose With Covariance From EKF Prediction: Equation (12) provides a way to transfer the confidence of pose estimation by scan–map registration, namely, uncertainty, from HNDT to EKF. Meanwhile, the predicted pose of the EKF is required by the LiDAR localization submodule as the initial pose estimation, namely, $\mathbf{X}$ in (7), which is generally obtained from the latest state prediction of EKF. In this study, both pose prediction $\mathbf{X}$ and its confidence matrix $\mathbf{P}$ are passed into the HNDT, and thus, the loss function $\mathrm{J}(\mathbf{X})$ in (9) is changed as follows:

$$\mathrm{J}(\mathbf{X}) = \sum_{j=1}^{N_p} d_1 \exp\left\{-\frac{d_2}{2}(\Delta\boldsymbol{p}_j)^{\mathrm{T}}\left(\boldsymbol{\Sigma}_j^{'}\right)^{-1}(\Delta\boldsymbol{p}_j)\right\} + \alpha\left((\mathbf{X}-\mathbf{X}_0)^{\mathrm{T}}\mathbf{P}^{-1}(\mathbf{X}-\mathbf{X}_0)\right) \tag{13}$$

where $\boldsymbol{\Sigma}_j^{'} = \mathbf{C}_j^{0.5}\mathbf{W}^2\mathbf{C}_j^{0.5}$. The solution of the HNDT-EKF tightly coupled algorithm is slightly different from HNDT in terms of the Hessian and Jacobian in (11), which are expressed as follows:

$$\left(\sum_j \mathbf{H}_j^k + \alpha\mathbf{P}^{-1}\right)\Delta\mathbf{X}^k = -\left(\sum_j \boldsymbol{g}_j^k + \alpha\mathbf{P}^{-1}(\mathbf{X}-\mathbf{X}_0)\right). \tag{14}$$

With the proposed two-way communication mechanism, HNDT and EKF are tightly connected, and they interact with each other using a covariance matrix. The interaction between the $\mathbf{R}$ matrix and $\mathbf{P}$ is shown in Fig. 4. During the prediction process, the covariance, namely, uncertainty, gradually increases. When the observation corrects the state, the uncertainty decreases. The value of matrix $\mathbf{R}$ is determined by the LiDAR localization results, and it reflects the uncertainty of the LiDAR pose.

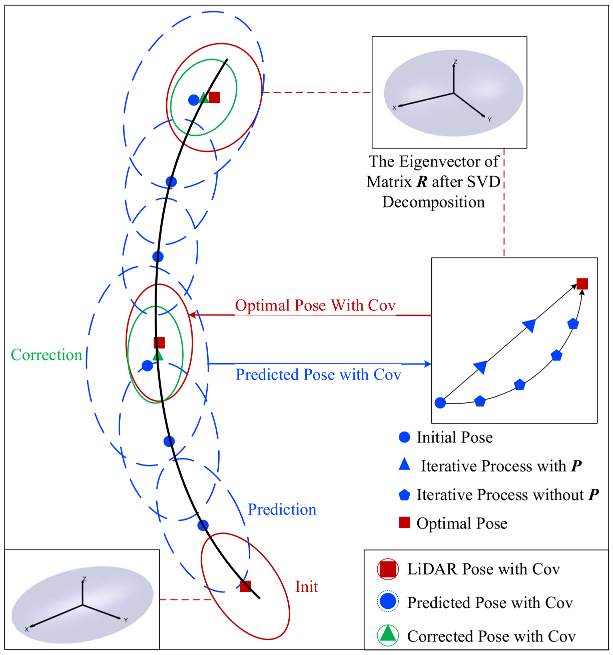


Fig. 4. Schematic of position covariance changes in the matrix $\boldsymbol{P}$ and $\boldsymbol{R}$.

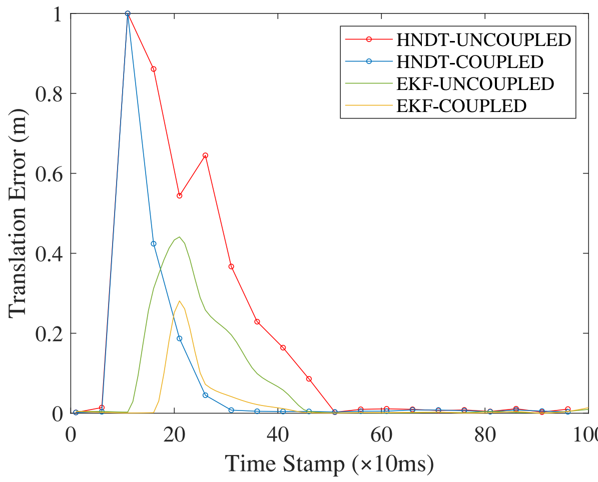


Fig. 5. Ablation experiment of HNDT-EKF tightly coupled and uncoupled algorithms.

To visualize the effect of our proposed HNDT-EKF tightly coupled algorithm, we added a one-meter error to the "X" direction of the LiDAR localization result at a certain moment and observed the localization results of HNDT and EKF in uncoupled and tightly coupled states, respectively, which are shown in Fig. 5. It is obvious that due to the effect of (12) and (13), HNDT and EKF algorithms are more robust to a sudden disturbance. The amplitude of the tightly coupled HNDT-EKF is smaller while its convergence speed is faster, which shows that the proposed algorithm is more robust to

TABLE II
NDT EXECUTION TIME COMPARISON BETWEEN FPGA AND CPU WITH PCL LIBRARY

Sub-module	Time Consumption (ms)	
	FPGA	CPU
Local Map Indexing	1.61	98.73
Registration	2.52	38.56
Others	5.67	86.86
Summary	9.80	224.15

external and internal noise interference. Through experiment, when $\alpha = 10$ and $\lambda = 1$, the algorithm could achieve the best result.

V. HARDWARE AND SOFTWARE CO-DESIGN

How can the localization framework be operated on a low-power hardware? Here, we evaluate the run time of each component of the NDT (using the PCL library) on the CPU, as shown in Table II. In a scan–map registration, computing time is mainly consumed by submap indexing, rigid transformation calculation, and other processes such as data preprocessing. All the major components of the NDT-based algorithm are time consuming, but the reasons are different. For map indexing, to effectively store a sparse map, different data structures, such as KD-Tree and hash map, are used, and redundant indexing overhead of these data structures will be added inevitably. For NDT computation, parallelism is restricted by both the CPU and memory bandwidth.

The problems are resolved by a software–hardware co-optimization approach on our computing platform. First, the map indexing algorithm from KD-TREE is changed to a hardware-friendly double-hash table. Similarly, the NDT computation module on the FPGA is developed to exploit parallelism. In general, map indexing can be accelerated from 98.73 to 1.61 ms with the map indexing algorithm proposed in Section IV-A, and the NDT computation algorithm is accelerated from 38.56 to 2.52 ms with CPU-FPGA heterogeneous architecture. The proposed localization framework has been deployed on an ARM-FPGA platform, where the FPGA performs as a registration algorithm accelerator. The process of algorithm execution on the FPGA is shown in Fig. 6. To maximize the efficiency of both the CPU and FPGA, operations are classified into "point-wise operations" and "once-for-iteration (OFI) operations." Point-wise operations are implemented on an FPGA with high parallelism, and OFI operations are executed on the CPU due to high clock frequency and flexibility.

Denoting the point cloud from the LiDAR real-time scan with $\mathcal{C} = \{\boldsymbol{c}_j | j = 1, 2, \ldots, N\}$ and the estimated vehicle pose at the ith iteration with $\mathbf{X}_i$, the NDT registration can be expressed using the following function:

$$\mathbf{X}_{i+1} = \text{NDT}(\mathbf{X}_i, \boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_N). \tag{15}$$

Using the MapReduce methodology, function NDT$(\cdot)$ can be separated into three subfunctions

$$\text{NDT}(\mathbf{X}_i, \boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_N) = f\left(\sum_j g\big(h(\mathbf{X}_i), \boldsymbol{c}_j\big)\right) \tag{16}$$

where $h(\cdot)$ performs pose-relevant but point-irrelevant operations, including Euler angle/rotation matrix conversion and partial derivatives to the vehicle pose; $g(\cdot)$ performs point-relevant calculation, including voxels' ND access and derivative computation for each point; and finally, the $f(\cdot)$ function synthesizes all derivatives from the whole points and generates the aligned pose.

Functions $h(\cdot)$ and $f(\cdot)$ are executed only once for every iteration, and hence, they are OFI operations. Because these OFI operations are applied to a few small-scale data, such as a 6-D vector for $h(\cdot)$, but have complex procedures such as QR decomposition, we deploy OFI operations on the CPU. In contrast, the function $g(\cdot)$ is a point-wise operation that is executed for every point in one iteration. Considering that the point-wise operation $g(\cdot)$ has no cross-dependencies between the execution on different points, we deploy it on an FPGA with a complex pipeline exploiting high parallelism.

VI. EXPERIMENTS

In order to verify the effectiveness of the proposed self-localization framework, experiments are executed on "Pioneer" autonomous vehicle equipped with a "XAZU3EG" chip on a hardware computing platform, and three LiDAR (VLP-32C$\times$1 and VLP-16$\times$2), which is shown in Fig. 1. And a public dataset (SouthBay dataset[1] [60]) is also utilized here, which could be divided into two main scenarios: 1) downtown and 2) highway. Owing to the lack of notable objects, the highway scene is the typical corner case for all LiDAR localization systems, and it will be ideal to validate the effectiveness of our improvements in Section IV. Therefore, two subsets (SanJoseDownTown and HighWay237) of the Apollo SouthBay dataset were used in this study.

To quantitatively analyze the performance of the self-localization algorithm, we define four metrics, including rotation error, translation error, latency, and localization loss rate. The rotation and translation error are assessed by the root-mean-square error (RMSE), where the rotation error compares the angular difference between the ground-truth rotation, and the translation error compares the position difference between the ground-truth position. The latency indicates the calculation time consumption. The location loss rate refers to the difference between the pose estimation and ground-truth pose exceeding a certain threshold, namely, mismatching between real-time scan and submap. The reason for this metric being specifically defined is that different from other kinds of localization methods, for all the self-localization system, once it happens to location loss, it is practically impossible to find the correct pose again, namely, kidnapped robot problem. Thus, a qualified self-localization system needs to maintain stably all the time in a large-scale urban environment test.

[1]https://apollo.auto/southbay.html

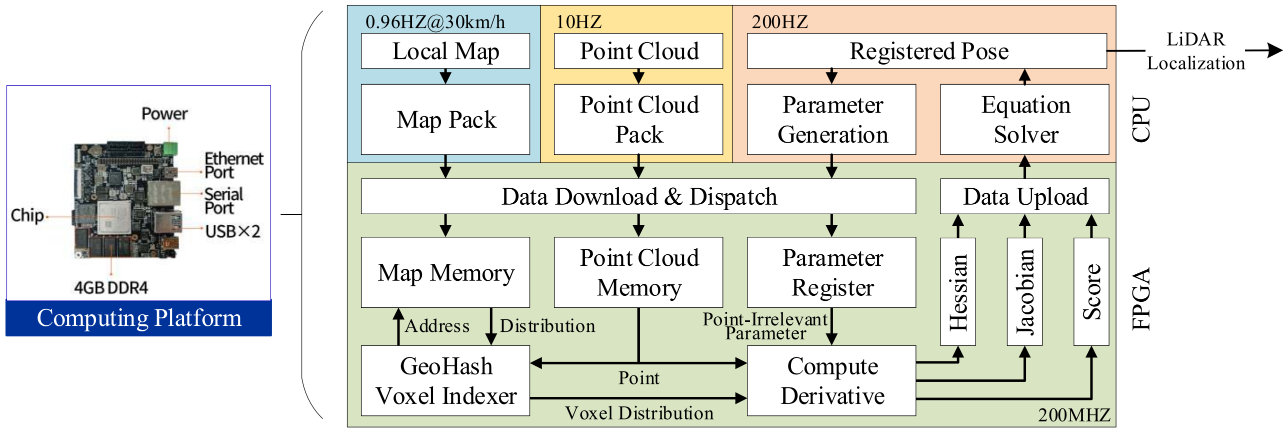


Fig. 6. CPU-FPGA heterogeneous computing system and data flow. It is adopted to accelerate the registration submodule, which contains the HNDT registration (red and green), auxiliary modules (blue), and data pack module (yellow).

Metrics are defined as follows.

1) Rotation Error (rad): $E_{\mathcal{R}} = \arccos((\mathrm{trace}(\mathbf{Rot}_{\mathcal{G}}\mathbf{Rot}^{\mathrm{T}}) - 1)/2)$.
2) Translation Error (m): $E_{\mathcal{T}} = ||\boldsymbol{t} - \boldsymbol{t}_{\mathcal{G}}||$.
3) Latency (ms): $L_t = t_{\text{out}} - t_{\text{in}}$.
4) Location loss rate (%): $\lambda = N_{\text{mis}}/N_{\text{sum}}$

where $\mathbf{Rot}_{\mathcal{G}}$ and $\boldsymbol{t}_{\mathcal{G}}$ are ground-truth rotation and translation, respectively. t_{in} refers to the timestamp of real-time scan received and t_{out} is the timestamp of the calculation result published. N_{mis} indicates the number of mismatching frames while N_{sum} is the total number of tested frames. The "mismatching" means the registration failure where $E_{\mathcal{T}} > 3.0$ or $E_{\mathcal{R}} > 0.7$.

A. Global Map Encoding and Local Map Update

In general, the global navigation map required by the autonomous vehicle localization framework should cover the entire drivable area of a city, even a country. Assuming that the data volume of the map is proportional to its area, a city (e.g., 10 km × 10 km) needs about 1.18 TB to store the point cloud navigation map, whereas only 0.496-GB storage is required by our hierarchical map.

As mentioned above, directly using the global map as the target point cloud will waste lots of computing resources and lead to registration divergence for the large difference between scan and map. Therefore, a suitable submap needs to be updated along with vehicle motion, and thus, quickly indexing to the corresponding submap according to the current vehicle location in a large-scale global map is the basis of real-time scan–map registration.

Through GeoHash encoding and double-hash mapping, efficient submap updating can be realized with rare latency and performance comparison with other methods, as shown in Table III, where different "element" means different minimum data units, "point cloud" means storing points directly, and "voxel" means storing statistics information of a voxel, whose resolution is 1.5 m × 1.5 m × 1.5 m. Finally, "ours" is a hierarchical map, in which valid blocks and voxels are included. In addition, the global map is a 1.3 km × 1.1 km domain, shown on the top left in Fig. 2(a). "Method" in Table III means the submap searching method. Different map elements determine their most efficient searching methods. It can be seen from the table that the time complexity of our hierarchical map is constant, which means that the submap searching time will not increase along with the scale of the global map increasing, and only the memory occupation for map storage will increase slightly.

TABLE III
COMPARISON OF DIFFERENT GLOBAL MAP STORING AND LOCAL MAP INDEXING METHOD

Element	Number	Occupation	Method	Complexity	L_t
Point	4.97×10^7	16.9 GB	KD-TREE	$O(log(n))$	290
Voxel	1.32×10^5	29.4 MB	Traverse	$O(n)$	13
Ours	1.03×10^3	**7.1MB**	Double hash	$O(1)$	**1.4**

B. LiDAR Localization

For the LiDAR localization submodule, experiments are implemented on SanJoseDownTown and HighWay237 of the Apollo SouthBay dataset, in which the mapping data is used for hierarchical map construction and the test data is used for localization performance evaluation. We projected all the frames into the world coordinate system according to their ground truth pose to generate a global navigation map. The submap is KNN (here $K = 50\,000$) of the global map according to each frame's initial position. In order to expedite the execution and improve the accuracy of all the algorithms, we filtered the ground points for both the source and target point clouds.

To verify the performance of the HNDT algorithm based on FPGA acceleration in Sections IV-B and V, it is compared with several other LiDAR localization methods, such as NDT-OMP [47], GICP-OMP [47], and VGICP-CUDA [48]. These methods are designed for accelerating LiDAR localization, and they are widely used for LiDAR localization on autonomous driving vehicles. The experimental results are shown in Table IV, where "DOW" and "HIG" represent the result in downtown and highway, respectively. Both NDT and HNDT algorithms based on FPGA acceleration, as shown in

TABLE IV
COMPARISON OF DIFFERENT REGISTRATION ACCELERATION METHODS

Method	Platform	Power (W)	Frame rate (HZ)		L_t		$E_{\mathcal{T}}$		$E_{\mathcal{R}}(\times 10^{-3})$		λ	
			DOW	HIG	DOW	HIG	DOW	HIG	DOW	HIG	DOW	HIG
NDT	XAZU3EG	**10**	20	20	39	34	0.29	0.33	6.21	7.13	0	0.97
HNDT	XAZU3EG	**10**	20	20	35	29	**0.16**	**0.24**	**3.69**	**5.78**	0	**0**
NDT-OMP [47]	i7-7820HQ	125	10.57	13.12	81	74	0.28	0.36	6.14	7.08	0	0.83
GICP-OMP [47]	i7-7820HQ	125	8.64	11.01	122	95	0.24	0.31	5.22	8.49	0.17	3.33
VGICP [48]	RTX2080	250	20	20	**23**	**18**	0.28	0.45	6.45	9.57	0.26	3.15

TABLE V
COMPLETE COMPARISON ON THE APOLLO SOUTHBAY DATASET

Area	Method	Horiz. RMS	<0.1m Pct.	Yaw. RMS	<0.1° Pct.
Baylands	Lu [42]	0.050	96.48%	**0.020**	**99.35**%
	Ding [61]	**0.041**	**99.05**%	0.042	90.16%
	Ours	0.048	96.91%	0.036	93.65%
ColumbiaPark	Lu [42]	**0.043**	98.02%	**0.028**	**99.50**%
	Ding [61]	0.046	95.79%	0.095	55.88%
	Ours	0.044	96.32%	0.045	92.54$
Highway237	Lu [42]	**0.045**	**99.01**%	0.038	**99.30**%
	Ding [61]	0.049	96.86%	**0.026**	98.40%
	Ours	0.059	92.13%	0.047	91.56%
MathildaAVE	Lu [42]	0.051	98.87%	**0.019**	**99.31**%
	Ding [61]	**0.048**	99.08%	0.076	75.36%
	Ours	0.051	95.61%	0.041	95.01%
SanjoseDowntown	Lu [42]	0.055	91.32%	0.034	**98.86**%
	Ding [61]	0.056	90.46%	0.045	91.03%
	Ours	**0.049**	**92.77**%	**0.031**	93.43%
SunnyvaleBigLoop	Lu [42]	0.055	92.42%	**0.033**	**96.44**%
	Ding [61]	0.075	78.72%	0.077	74.76%
	Ours	**0.053**	**92.86**%	0.034	91.87%

the table, reach the maximum frame rate (20 Hz), which is the same performance as VGICP accelerated by GPU, whereas the power of our computing platform is only one-twentieth of a GPU. In addition, the performance of the HNDT in terms of accuracy and robustness is the best one. Compared with downtown, the scenes of highways are sparser, and thus, the performance of all algorithms will decrease. However, HNDT exhibits the least performance degradation among all the algorithms, and there is no mismatch (divergence) in all the experiments, which shows that our homogeneous method can effectively improve the robustness and accuracy of the registration algorithm in sparse scenes. The complete experimental results on the Apollo SouthBay dataset are shown in Table V, in which the evaluation metrics are consistent with the original paper [42].

In the Apollo SouthBay dataset, the translation error between the initial pose and ground truth is large enough, but the rotation error is too small, which makes the rotation error in the estimated pose of the registration algorithm is almost negligible. In other words, even if only the translation part is calculated in the LiDAR localization submodule and the rotation part remains the initial value, quite good pose estimation results can be obtained. Therefore, in order to better demonstrate the improvement of the proposed algorithms, we use different initial poses for LiDAR localization estimation. Here, several classic and efficient registration algorithms, including ICP [34], AA-ICP [46], and CPD [40], are introduced for comparison. Note that there is no available acceleration method for the three algorithms mentioned above, so experiments are implemented with C++ and tested offline on a PC with an Intel Core i7 2.9-GHz CPU.

The experimental results are presented in Fig. 7. We can note that the accuracy of the proposed HICP and HNDT is higher than those of other algorithms, such as GICP and AAICP, under different translation and rotation noise. Fig. 7(c) shows that there are no outliers (registration failure or divergence) in HNDT and HICP, which demonstrates that our proposed homogeneous method in (5) and (9) can improve the robustness of the registration algorithm, especially with inaccurate initial pose estimation. It is not difficult to interpret that local point cloud distribution information could supply a more accurate gradient compared with the P2P method, and the entirety of the distribution information [proposed in (5)] makes the iteration step more homogeneous in all directions. Moreover, the accuracy of HICP is slightly higher than that of HNDT, as shown in Fig. 7(a) and (b). Because the local distribution is generated by KNNS in HICP, but for HNDT, it is generated by voxel division, which can be considered as an approximation of KNN. Finally, from the perspective of time consumption, HNDT consumes lesser time compared with all other algorithms, whereas HICP takes longer than the original ICP and AAICP but shorter than the GICP. To sum up, HICP could also obtain the best result in terms of accuracy and robustness on the two datasets, but the time consumption is intolerable, whereas HNDT has comparable accuracy and robustness with the shortest time consumption.

C. HNDT-EKF Tightly Coupled Localization

The Apollo SouthBay dataset does not provide the vehicle kinematics model and other necessary data such as wheel speed and IMU data, so it is impossible to implement the HNDT-EKF tightly coupled algorithm on it; thus, the complete localization framework test is performed on our Pioneer autonomous driving vehicle in real world scene. The ground truth of the vehicle pose was generated by a high-precision INS (Novatel Pwrpak7, IMU-ISA-100C).

We conducted tests on the Pioneer autonomous vehicle in urban and highway scenes to verify the improvement of our

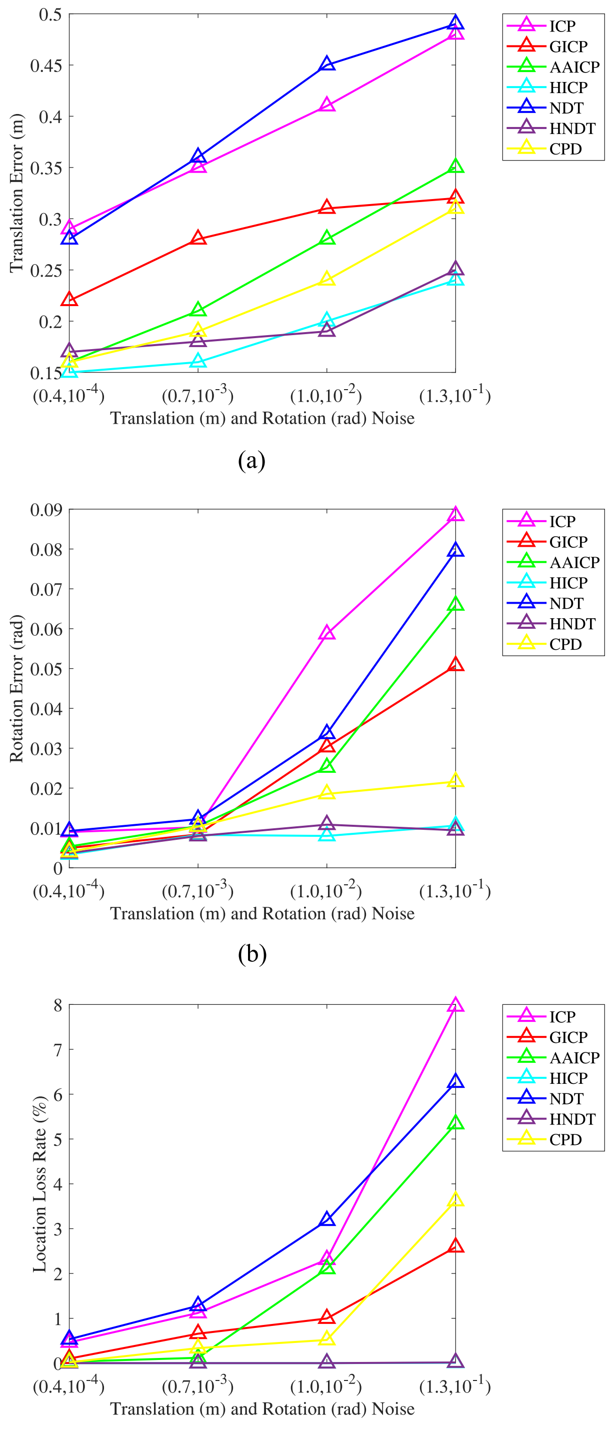


Fig. 7. Rotation and translation error under different noise. Each two-tuple of the X-label in figures represent translation (unit: m) and rotation (unit: rad) noise, respectively. (a) Translation error. (b) Rotation error. (c) Location loss rate.

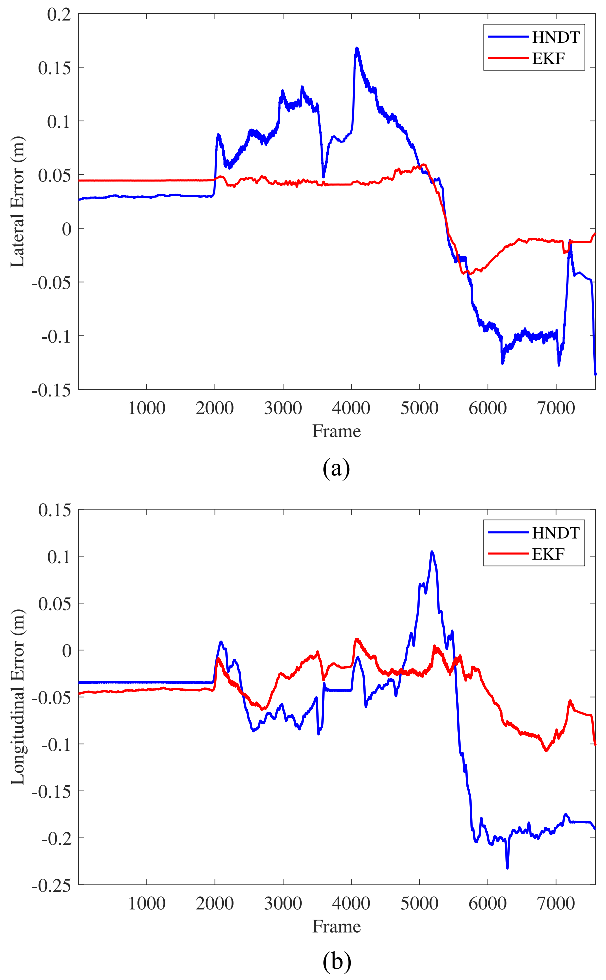


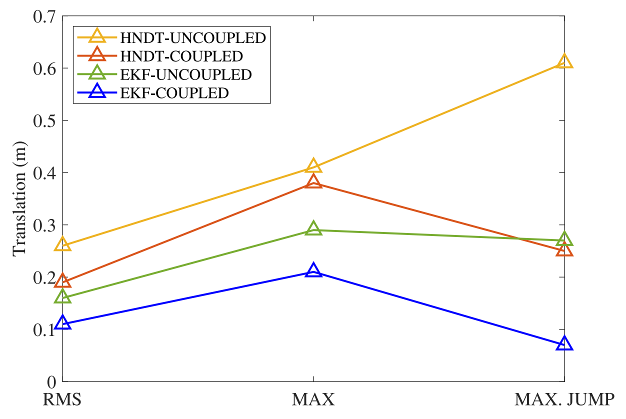
Fig. 8. (a) Latitudinal and (b) longitudinal errors during localization.

self-localization framework using the HNDT-EKF tightly coupled algorithm. The latitudinal and longitudinal errors of the different methods resulting from localization are shown in Fig. 8, in which the latitude and longitude errors are between vehicle position estimated by the proposed self-localization framework and INS. During the experiment, the vehicle speed is about 40 km/h. The first 2000 frames of the route consist of an urban road with buildings and trees on both sides. Proceeding from the 2000th frame, the vehicle enters a highway, where only light poles and low fences can provide structure information for registration. Thus, there is some jitter in the localization result. But the accuracy of the HNDT-EKF remains within 0.1 m.

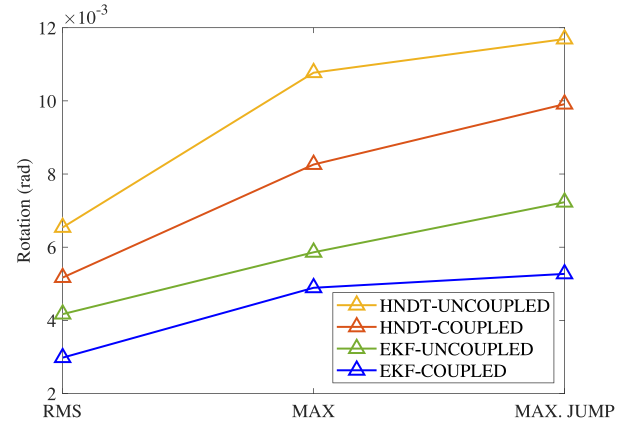
Another set of ablation experiments is conducted to compare HNDT-EKF tightly coupled and uncoupled algorithms. There are two important metrics, including maximum error and jump, used to describe smoothness. Assuming the translation error of the previous frame is -5 cm and the next frame is 5 cm, then the jump step is 10 cm. The results are shown in Fig. 9. The proposed tightly coupled HNDT-EKF algorithm effectively reduces the maximum error and the maximum jump compared with uncoupled algorithms. Benefitting from (12) and (13), the proposed HNDT-EKF tight-coupled algorithm is more robust with higher precision.

VII. Conclusion

This article proposes a self-localization framework to solve the localization drift problem caused by poor GPS signal for autonomous vehicles in urban environments. It can achieve



(a)



(b)

Fig. 9. Statistics information of experiment results on highway and urban scenes. (a) Translation. (b) Rotation.

high-precision and robust positioning results based on onboard sensors and a hierarchical map, which makes underground parking lots, urban highway, and other GPS-denied areas into drivable areas. Different from other map-based localization methods that relying on a high-performance computing platform to accelerate the scan–map registration, the proposed self-localization framework is implemented on a low-power ARM+FPGA platform.

We have improved the performance of the framework from three aspects: 1) global map multiresolution encoding; 2) LiDAR localization; and 3) multisensor fusion, and experiments on different datasets proved the effectiveness of our proposed algorithm. Through software and hardware co-design, the parallelism of the framework is greatly improved, so that the latency of the framework is significantly reduced, and the power consumption is only one-tenth that of other acceleration algorithms, which makes it more friendly to autonomous driving vehicles in large-scale urban scenes.

References

[1] N.-N. Zheng *et al.*, "Hybrid-augmented intelligence: Collaboration and cognition," *Front. Inf. Technol. Electron. Eng.*, vol. 18, no. 2, pp. 153–179, 2017.

[2] F.-Y. Wang *et al.*, "China's 12-year quest of autonomous vehicular intelligence: The intelligent vehicles future challenge program," *IEEE Intell. Transp. Syst. Mag.*, vol. 13, no. 2, pp. 6–19, Mar. 2021.

[3] L. Li, N.-N. Zheng, and F.-Y. Wang, "On the crossroad of artificial intelligence: A revisit to Alan Turing and Norbert Wiener," *IEEE Trans. Cybern.*, vol. 49, no. 10, pp. 3618–3626, Oct. 2019.

[4] S. Chen, Z. Jian, Y. Huang, Y. Chen, Z. Zhou, and N. Zheng, "Autonomous driving: Cognitive construction and situation understanding," *Sci. China Inf. Sci.*, vol. 62, no. 8, pp. 1–27, 2019.

[5] S. Chen, S. Zhang, J. Shang, B. Chen, and N. Zheng, "Brain-inspired cognitive model with attention for self-driving cars," *IEEE Trans. Cogn. Develop. Syst.*, vol. 11, no. 1, pp. 13–25, Mar. 2019.

[6] J. Roche, V. De-Silva, and A. Kondoz, "A multimodal perception-driven self evolving autonomous ground vehicle," *IEEE Trans. Cybern.*, early access, Oct. 8, 2021, doi: 10.1109/TCYB.2021.3113804.

[7] M. Deng, Z. Li, Y. Kang, C. L. P. Chen, and X. Chu, "A learning-based hierarchical control scheme for an exoskeleton robot in human–robot cooperative manipulation," *IEEE Trans. Cybern.*, vol. 50, no. 1, pp. 112–125, Jan. 2020.

[8] R.-E. Precup, T.-A. Teban, A. Albu, A.-B. Borlea, I. A. Zamfirache, and E. M. Petriu, "Evolving fuzzy models for prosthetic hand myoelectric-based control," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 7, pp. 4625–4636, Jul. 2020.

[9] S. Feng and C. L. P. Chen, "Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 414–424, Feb. 2020.

[10] A. Turnip and J. H. Panggabean, "Hybrid controller design based magneto-rheological damper lookup table for quarter car suspension," *Int. J. Artif. Intell.*, vol. 18, no. 1, pp. 193–206, 2020.

[11] T. Haidegger, L. Kovács, R.-E. Precup, S. Preitl, B. Benyó, and Z. Benyó, "Cascade control for telerobotic systems serving space medicine," *IFAC Proc. Vol.*, vol. 44, no. 1, pp. 3759–3764, 2011.

[12] C. Pozna, F. Troester, R.-E. Precup, J. K. Tar, and S. Preitl, "On the design of an obstacle avoiding trajectory: Method and simulation," *Math. Comput. Simul.*, vol. 79, no. 7, pp. 2211–2226, 2009.

[13] U. Yuhana, N. Z. Fanani, E. M. Yuniarno, S. Rochimah, L. T. Koczy, and M. H. Purnomo, "Combining fuzzy signature and rough sets approach for predicting the minimum passing level of competency achievement," *Int. J. Artif. Intell.*, vol. 18, no. 1, pp. 237–249, 2020.

[14] Z. Preitl, R.-E. Precup, J. K. Tar, and M. Takács, "Use of multi-parametric quadratic programming in fuzzy control systems," *Acta Polytechnica Hungarica*, vol. 3, no. 3, pp. 29–43, 2006.

[15] H. Huang, C. Yang, and C. L. P. Chen, "Optimal robot–environment interaction under broad fuzzy neural adaptive control," *IEEE Trans. Cybern.*, vol. 51, no. 7, pp. 3824–3835, Jul. 2021.

[16] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot. Sci. Syst.*, vol. 2, 2014, pp. 1–9.

[17] T. Botterill, S. Mills, and R. Green, "Correcting scale drift by object recognition in single-camera SLAM," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1767–1780, Dec. 2013.

[18] M. U. M. Bhutta, M. Kuse, R. Fan, Y. Liu, and M. Liu, "Loop-box: Multiagent direct SLAM triggered by single loop closure for large-scale mapping," *IEEE Trans. Cybern.*, early access, Nov. 6, 2020, doi: 10.1109/TCYB.2020.3027307.

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

[20] B. Guan, J. Zhao, Z. Li, F. Sun, and F. Fraundorfer, "Relative pose estimation with a single affine correspondence," *IEEE Trans. Cybern.*, early access, Apr. 28, 2021, doi: 10.1109/TCYB.2021.3069806.

[21] S. Chen, Y. Chen, S. Zhang, and N. Zheng, "A novel integrated simulation and testing platform for self-driving cars with hardware in the loop," *IEEE Trans. Intell. Veh.*, vol. 4, no. 3, pp. 425–436, Sep. 2019.

[22] P. D. Groves, "Principles of GNSS, inertial, and multisensor integrated navigation systems, 2nd edition [Book review]," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 30, no. 2, pp. 26–27, Feb. 2015.

[23] W. Wen, G. Zhang, and L.-T. Hsu, "Correcting NLOS by 3D LiDAR and building height to improve GNSS single point positioning," *Navigation*, vol. 66, no. 4, pp. 705–718, 2019.

[24] K.-H. Lam, C.-C. Cheung, and W.-C. Lee, "RSSI-based LoRa localization systems for large-scale indoor and outdoor environments," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 11778–11791, Dec. 2019.

[25] K. Guo, X. Li, and L. Xie, "Ultra-wideband and odometry-based cooperative relative localization with application to multi-UAV formation control," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2590–2603, Jun. 2019.

[26] J. Wang, N. Tan, J. Luo, and S. J. Pan, "WOLoc: WiFi-only outdoor localization using crowdsensed hotspot labels," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Atlanta, GA, USA, 2017, pp. 1–9.

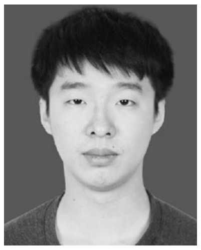
[27] W. Kim, J. Park, J. Yoo, H. J. Kim, and C. G. Park, "Target localization using ensemble support vector regression in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 43, no. 4, pp. 1189–1198, Aug. 2013.
[28] L. Yin, Q. Ni, and Z. Deng, "Intelligent multisensor cooperative localization under cooperative redundancy validation," *IEEE Trans. Cybern.*, vol. 51, no. 4, pp. 2188–2200, Apr. 2021.
[29] B. Yang, Q. Qiu, Q.-L. Han, and F. Yang, "Received signal strength indicator-based indoor localization using distributed set-membership filtering," *IEEE Trans. Cybern.*, vol. 52, no. 2, pp. 727–737, Feb. 2022.
[30] I. A. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun, "Learning to localize using a LiDAR intensity map," 2020, *arXiv:2012.10902*.
[31] K. Yoneda, H. Tehrani, T. Ogawa, N. Hukuyama, and S. Mita, "Lidar scan feature for localization with highly precise 3-D map," in *Proc. IEEE Intell. Veh. Symp.*, Dearborn, MI, USA, 2014, pp. 1345–1350.
[32] L. Wang, Y. Zhang, and J. Wang, "Map-based localization method for autonomous vehicles using 3D-LiDAR," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 276–281, 2017.
[33] P. Egger, P. V. K. Borges, G. Catt, A. Pfrunder, R. Siegwart, and R. Dubé, "PoseMap: Lifelong, multi-environment 3D LiDAR localization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Madrid, Spain, 2018, pp. 3430–3437.
[34] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, 1992, pp. 239–256, doi: 10.1109/34.121791.
[35] C. Dorai, G. Wang, A. K. Jain, and C. Mercer, "Registration and integration of multiple object views for 3D model construction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 83–89, Jan. 1998.
[36] R. Benjemaa and F. Schmitt, "Fast global registration of 3D sampled surfaces using a multi-z-buffer technique," *Image Vis. Comput.*, vol. 17, no. 2, pp. 113–123, 1999.
[37] Y. Yang, D. Fan, S. Du, M. Wang, B. Chen, and Y. Gao, "Point set registration with similarity and affine transformations based on bidirectional KMPE loss," *IEEE Trans. Cybern.*, vol. 51, no. 3, pp. 1678–1689, Mar. 2021.
[38] M. Magnusson, "The three-dimensional normal-distributions transform—An efficient representation for registration, surface analysis, and loop detection," Ph.D. dissertation, Dept. Rebot. Autom. Process Control, Örebro universitet, Örebro, Sweden, 2009.
[39] L. Jun, L. Wei, D. Donglai, and S. Qiang, "Point cloud registration algorithm based on NDT with variable size voxel," in *Proc. 34th Chin. Control Conf.*, Hangzhou, China, 2015, pp. 3707–3712.
[40] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, Dec. 2010.
[41] L. Li, M. Yang, C. Wang, and B. Wang, "Robust point set registration using signature quadratic form distance," *IEEE Trans. Cybern.*, vol. 50, no. 5, pp. 2097–2109, May 2020.
[42] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song, "DeepVCP: An end-to-end deep neural network for point cloud registration," in *Proc. IEEE Int. Conf. Comput. Vis.*, Seoul, South Korea, 2019, pp. 12–21.
[43] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proc. IEEE Int. Conf. Comput. Vis.*, Seoul, South Korea, 2019, pp. 3523–3532.
[44] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. Robot. Sci. Syst.*, vol. 2. Seattle, WA, USA, 2009, p. 435.
[45] T. Stoyanov, M. Magnusson, H. Andreasson, and A. J. Lilienthal, "Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations," *Int. J. Robot. Res.*, vol. 31, no. 12, pp. 1377–1393, 2012.
[46] A. L. Pavlov, G. W. Ovchinnikov, D. Y. Derbyshev, D. Tsetserukou, and I. V. Oseledets, "AA-ICP: Iterative closest point with Anderson acceleration," in *Proc. IEEE Int. Conf. Robot. Autom.*, Brisbane, QLD, Australia, 2018, pp. 1–6.
[47] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LiDAR-based system for long-term and wide-area people behavior measurement," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, 2019, Art. no. 1729881419841532.
[48] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized GICP for fast and accurate 3D point cloud registration," in *Proc. IEEE Intern. Conf. Robot. Autom. (ICRA)* 2021, pp. 11054–11059, doi: 10.1109/ICRA48506.2021.9560835
[49] R. W. Wolcott and R. M. Eustice, "Robust LIDAR localization using multiresolution Gaussian mixture maps for autonomous driving," *Int. J. Robot. Res.*, vol. 36, no. 3, pp. 292–319, 2017.
[50] R. W. Wolcott and R. M. Eustice, "Fast LIDAR localization using multiresolution Gaussian mixture maps," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seattle, WA, USA, 2015, pp. 2814–2821.
[51] N. Pous, D. Gingras, and D. Gruyer, "Intelligent vehicle embedded sensors fault detection and isolation using analytical redundancy and nonlinear transformations," *J. Control Sci. Eng.*, vol. 2017, Jan. 2017, Art. no. 1763934.
[52] F. Gustafsson *et al.*, "Particle filters for positioning, navigation, and tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 425–437, Feb. 2002.
[53] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing," *Robot. Atuon. Syst.*, vol. 61, no. 8, pp. 721–738, 2013.
[54] W. Wen, T. Pfeifer, X. Bai, and L.-T. Hsu, "It is time for factor graph optimization for GNSS/INS integration: Comparison between FGO and EKF," 2020, *arXiv:2004.10572*.
[55] A. Assa and F. Janabi-Sharifi, "A robust vision-based sensor fusion approach for real-time pose estimation," *IEEE Trans. Cybern.*, vol. 44, no. 2, pp. 217–227, Feb. 2014.
[56] J. Li, J. Bao, and Y. Yu, "Study on localization for rescue robots based on NDT scan matching," in *Proc. IEEE Int. Inf. Autom.*, Harbin, China, 2010, pp. 1908–1912.
[57] N. Akai *et al.*, "Autonomous driving based on accurate localization using multilayer LiDAR and dead reckoning," in *Proc. IEEE Intell. Transp. Syst.*, Yokohama, Japan, 2017, pp. 1–6.
[58] Y. Shen, C. Xia, Z. Jian, S. Chen, and N. Zheng, "An integrated localization system with fault detection, isolation and recovery for autonomous vehicles," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Indianapolis, IN, USA, 2021, pp. 84–91.
[59] Y. Yang, C. Xia, X. Deng, Y. Shen, S. Chen, and N. Zheng, "HeLPS: Heterogeneous LiDAR-based positioning system for autonomous vehicle," in *Proc. IEEE 46th Annu. Conf. Ind. Electron. Soc.*, Singapore, 2020, pp. 618–625.
[60] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song, "L3-net: Towards learning based LiDAR localization for autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Long Beach, CA, USA, 2019, pp. 6389–6398.
[61] W. Ding, S. Hou, H. Gao, G. Wan, and S. Song, "LiDAR inertial odometry aided robust LiDAR localization system in changing city scenes," in *Proc. IEEE Int. Conf. Robot. Autom.*, Paris, France, 2020, pp. 4322–4328.

Chao Xia received the B.E. degree in electronic and information engineering from Xi'an Jiaotong University, Xi'an, China, in 2017, where he is currently pursuing the Ph.D. degree with the Institute of Artificial Intelligence and Robotics.

His research interests mainly include autonomous vehicle localization, point cloud registration and semantic segmentation, and deep learning.

Yanqing Shen received the B.E. degree in electronic and information engineering from Xi'an Jiaotong University, Xi'an, China, in 2019, where she is currently pursuing the Ph.D. degree with the Institute of Artificial Intelligence and Robotics.

Her research interests mainly include autonomous vehicle localization, semantic visual SLAM, and cognitive understanding.



Yuedong Yang received the B.E. degree in electronic and information engineering from Xi'an Jiaotong University, Xi'an, China, in 2017, where he is currently pursuing the Ph.D. degree with the Institute of Artificial Intelligence and Robotics.

His research interests mainly include edge computation and hardware–software co-design.

Xiaodong Deng is currently pursuing the B.E. degree in electronic and information engineering with Xi'an Jiaotong University, Xi'an, China.

His research interests mainly include autonomous driving, computer vision, and deep learning.

Shitao Chen (Member, IEEE) received the B.E. and Ph.D. degrees in electronic and information engineering from Xi'an Jiaotong University, Xi'an, China, in 2015 and 2021, respectively.

He is currently an Assistant Professor with Xi'an Jiaotong University. He is the Executive Vice President of Shunan Academy of Artificial Intelligence, Ningbo, China. He has published over 30 papers in various journals and conference proceedings, while many of them won the best paper awards in summits. His current research interests include computer vision, artificial intelligence, robotics, and autonomous vehicle.

Dr. Chen serves as the Editor or a Reviewer for IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON INTELLIGENT VEHICLES, IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS, ICRA, and other well-known international academic journals and conferences.

Jingmin Xin (Senior Member, IEEE) received the B.E. degree in information and control engineering from Xi'an Jiaotong University, Xi'an, China, in 1988, and the M.S. and Ph.D. degrees in electrical engineering from Keio University, Yokohama, Japan, in 1993 and 1996, respectively.

From 1988 to 1990, he was with the Tenth Institute of Ministry of Posts and Telecommunications of China, Xi'an. He was with Communications Research Laboratory, Tokyo, Japan, as an Invited Research Fellow of the Telecommunications Advancement Organization of Japan from 1996 to 1997 and as a Postdoctoral Fellow with the Japan Science and Technology Corporation, Saitama, Japan, from 1997 to 1999. He was also a Guest (Senior) Researcher with YRP Mobile Telecommunications Key Technology Research Laboratories Company Ltd., Yokosuka, Japan, from 1999 to 2001. From 2002 to 2007, he was with Fujitsu Laboratories Ltd., Yokosuka. Since 2007, he has been a Professor with Xi'an Jiaotong University. His research interests are in the areas of adaptive filtering, statistical and array signal processing, system identification, and pattern recognition.

Nanning Zheng (Fellow, IEEE) received the Graduate degree in electrical engineering and the M.S. degree in information and control engineering from Xi'an Jiaotong University, Xi'an, China, in 1975 and 1981, respectively, and the Ph.D. degree in electrical engineering from Keio University, Yokohama, Japan, in 1985.

He is a Distinguished Professor with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, where he is the Founder of the Institute of Artificial Intelligence and Robotics and is currently a Professor and the Director of the Institute of Artificial Intelligence and Robotics. His research interests include computer vision, pattern recognition, autonomous vehicle, and brain-inspired computing.

Prof. Zheng became a member of the Chinese Academy of Engineering in 1999. He is a Council Member of the International Association for Pattern Recognition.