2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)
October 8-12, 2022, Macau, China

Location Learning for AVs: LiDAR and Image Landmarks Fusion Localization with Graph Neural Networks

Qiyu Kang,[1] Rui She,[1] Sijie Wang,[1] Wee Peng Tay,[1]
Diego Navarro Navarro,[2] and Andreas Hartmannsgruber[2]

2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC) | 978-1-6654-6880-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITSC55140.2022.9922179

***Abstract*— High-accuracy vehicular self-localization plays an important role in autonomous driving. In this paper, we investigate the problem of estimating an autonomous vehicle's location using computer vision and LiDAR information, with respect to (w.r.t.) a reference map composed of landmarks from the environment. The map is generated off-line using static roadside objects such as traffic signs, traffic lights and roadside poles, which are organized into a graph for calibration. We use deep learning techniques to perform automatic feature extraction from sensor measurements. Specifically, we use a Convolutional Neural Network (CNN) to extract features from RGB images captured by an on-vehicle camera and use a Graph Neural Network (GNN) to integrate measurements from LiDAR scans. The vehicle's location is estimated from a regression neural network by comparing the extracted features from the real-time measurements with the calibration landmark map. In our experiments, we perform evaluations using 2 datasets and demonstrate that our approach achieves the state-of-the-art localization accuracy.**

***Index Terms*— Learning-based localization, Autonomous Driving, GNN, Image, LiDAR and Landmark.**

I. Introduction

In autonomous driving, a precisely estimated vehicle position plays a crucial role in the perception, planning, and navigation for autonomous vehicles (AVs) [1]. In structured or open environments, Global Navigation Satellite Systems (GNSSs), such as the Global Positioning System (GPS) or BeiDou Navigation Satellite System (BDS), usually provide sufficiently accurate localization results for autonomous driving tasks. However more accurate localization for autonomous vehicles (AVs) in unreliable GNSS scenarios, e.g. urban or tunnel areas where satellite signals are weakened or blocked, is required.

Although more advanced GNSS technologies like Differential GPS (DGPS) [1] and Real Time Kinematic (RTK) [2] may mitigate the issue by improving the localization accuracy compared to the conventional GPS, they have higher costs and still suffer from limited signal coverage. Therefore, GNSS cannot be used alone and is often integrated with other sensors in the context of autonomous driving. The inertial navigation system in AVs typically includes Inertial Motion Unit (IMU) sensors and wheel odometry sensors, which are not affected by the satellite signal coverage. The linear accelerations and vehicle angular velocities measured from accelerometers and gyroscopes in a IMU, together with the speed and turning measurements from the wheel odometry sensors, can be used to estimate the vehicle position relative to its initial position using a path integration technique as dead reckoning [3]. This method has low cost in vehicle localization system, but suffers from accumulated errors and sensor drifts even with advanced algorithms [4]. The magnitude of the localization errors can become too large to be useful for autonomous vehicles. Therefore, it is valuable to investigate autonomous driving localization methods that can achieve better accuracy and are more stable. For modern vehicle localization systems, other on-board sensors including cameras, LiDAR or Radar sensors are commonly equipped in AVs to deal with the aforementioned accumulated errors [5]–[7].

LiDAR sensors have been widely used in many localization works [5], [6] and they have good accuracy for measuring the ranges of targets in the environment. However, LiDAR sensors are weak in recognizing targets [8], which is one of the advantages of computer vision. Thus, in many other works, camera images are utilized to augment LiDAR measurments. In [9], the authors use only a single monocular camera to conduct ego localization. The camera image is used to estimate the ego position relative to a visual map previously computed. However, cameras cannot provide high-quality range information and their performance is influenced by light and weather conditions [8]. Hence, in [6], both LiDAR and RGB-depth camera are used for localization by incorporating visual tracking algorithms, depth information of the structured light sensor, and a low-level vision-LiDAR fusion algorithm. In [7], IMU, camera, and LiDAR are fused to realize three-dimensional localization. In this paper, we propose a new localization method using both camera and LiDAR sensors based on a graph neural network (GNN) data association approach. To the best of our knowledge, the use of graph related learning in vision-LiDAR localization has never been investigated before.

In the localization task, to avoid ambiguity and obtain better results, typically a reference global or local map [10] need to be first defined. To achieve robust localization, in [11], maps are regarded as probability distributions over

*First two authors contributed equally to this work.

This work is supported under the RIE2020 Industry Alignment Fund-Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s).

[1] The authors are with the Continental-NTU Corporate Lab, Nanyang Technological University, 50 Nanyang Avenue, 639798, Singapore. Emails: {kang0080@e.; rui.she@; wang1679@e.;wptay@}ntu.edu.sg

[2] The authors are with Continental Automotive Singapore Pte. Ltd., 80 Boon Keng Road, 339780, Singapore. Emails: {diego.navarro.navarro; andreas.hartmannsgruber}@continental.com.

environments in each cell. The accuracy of these methods exceeds that of GPS-based ones by over an order of magnitude. In [12], the authors use LiDAR sensors to obtain a pre-mapped environment, from which landmarks are extracted. The authors in [5] propose to use maximally stable extremal regions (MSER) to detect road markings recurrently. Different from all the above methods, in this paper, we use the semantic segmentation network DeepLabV3+ [13] to process the camera images to extract useful static landmark objects including traffic lights, traffic signs, and poles. These landmark patches together with the LiDAR point clouds projected on them are registered to be the nodes in a graph topological reference map in an off-line manner.

Given a reference map for calibration, the localization task aims to determine vehicle position using the real-time measurements from on-board sensors. In some works like [5], the measurements need not to be explicitly associated with the landmarks stored in the map. However, in many other works, data association or matching (i.e., the process of associating a measurement or feature when a vehicle transverses the environment to its corresponding previously extracted feature) is important. In [1], the authors use Pearson product-moment correlation to perform the data association. In visual images, Sum of Square Differences (SSD) and Normalized Cross Correlation (NCC) are traditional similarity indices that use the intensity differences between corresponding pixels in two image patches.

In this work, we perform the data association using Graph Attention Networks (GAT). The output of the network is the similarity between two image patches. After the data association, we compare the features from each matched pair to estimate the vehicle locations with respect to (w.r.t.) the reference map. More specifically, we use Resnet [14] to extract features from visual image patches and PointNet [15] to extract features from LiDAR points over the patches. The vehicle's location is estimated from MLP regression layers by comparing the matched pairs of extracted features from the equipped sensors' real-time measurements and the calibration landmark map.

Our main contributions are summarized as follows:

1) We introduce a new landmark map technique where roadside static landmark patches and the LiDAR points projected on them are organized in a graph for calibration.
2) We propose a learning-based localization method for AVs that consists of the localization neural network (LNN) and the landmark objects matching neural network (LOMNN), which is based on graph attention mechanisms.
3) We conduct comparisons to demonstrate that our learning-based localization framework achieves the state-of-the-art performance for the vehicular self-localization task.

The rest of this paper is organized as follows. In Section II, we give a detailed account of related work. We present our proposed vehicular self-localization framework in Section III. In Section IV, we evaluate the performance of our model on two datasets, with comparison to other baseline models. We conclude the paper in Section V.

II. RELATED WORK

In traditional multi-model localization systems, data fusion is required to perform the localization task with less uncertainty and better accuracy results compared to the case where those sources are used individually. In the literature, filtering methods like Kalman filter and its extensions including the Extended Kalman Filter (EKF) [16] and Sigma-Point Kalman Filters (SPKF) [17] are commonly used. The recursive estimation process allows the probabilistic descriptions of observations from different sensor models to be included in the Bayes update. In general, complex image data from camera sensors or the point cloud data from LiDAR scanners are inefficient to be directly utilized in the filtering methods even though they provide rich information. Traditionally, feature extraction techniques, e.g., SURF [18] and ORB [19], are first employed to extract useful information from the raw sensor data. By contrast, in deep learning approaches, neural networks with different architectures are applied directly to images or point clouds to exact features of interest to regress the vehicle movement [5], [9]. Similar to [5], we choose the learning approaches [14], [15] since they are more robust to environmental noise like illumination changes.

In the literature of landmark map construction, [20] uses Edge Boxes [21] to detect a bounding box (bbox) around a patch that contains a larger number of internal contours compared to the number of contours exiting from the box, which indicates the presence of an intelligible object in the enclosed patch. In [5], the authors propose to use maximally stable extremal regions (MSER) to detect road markings recurrently. The aforementioned patch extraction or landmark detection approaches are not stable enough for removing dynamic objects and many noisy regions are presented. By contrast, in our datasets, we use DeepLabV3+ [13] to obtain stable semantic segmentation masks for the camera images. We choose the pixels that are labelled as static roadside landmark objects, including traffic lights, traffic signs, and poles. In [22], the authors also use DeepLabV3+ to extract significant landmark regions. The difference between our landmark map construction and theirs is that we have discarded other temporarily static objects like parked cars, which are kept in [22] since that paper focuses on the place recognition problem.

III. FRAMEWORK

In this section, we detail our framework pipeline where crucial concepts, the localization neural network (LNN), and the landmark objects matching neural network (LOMNN) are presented. The key notations are summarized in Table I.

A. Coordinate Systems

In our localization framework, there exist three coordinate systems: the world coordinate system, the localization coordinate system, and the vehicle coordinate system. In this paper, we take the Universal Transverse Mercator (UTM)

TABLE I

SUMMARY OF KEY NOTATIONS

Notations	Explanations
L, r	The maximum road partition length and rotation.
$\mathcal{G}^m = (V^m, E^m)$	A graph topological landmark map in road partition m.
$s^{m,j}$	The j-th ($j \in \mathbb{N}^+$) landmark patch in partition m.
$p_{s^{m,j}}$	The points cloud reflected from landmark patch $s^{m,j}$.
$d^{m,j} = (s^{m,j}, p_{s^{m,j}})$	The j-th landmark object in partition m.
$\mathcal{G}_t^x = (V_t^x, E_t^x)$	A measurement graph at time t.
$\mathcal{G}_t^y$	A graph topological landmark map at time t ($= \mathcal{G}^m$ if the AV is in partition m).
$s_t^{x,i}$	The i-th ($i \in \mathbb{N}^+$) landmark patch in real-time camera frame captured at time t.
$p_{s_t^{x,i}}$	The points cloud reflected from the landmark patch $s_t^{x,i}$.
$d_t^{x,i} = (s_t^{x,i}, p_{s_t^{x,i}})$	The i-th landmark object in real-time camera frame captured at time t.
$\tilde{\mathcal{G}}_t^{x,i} = (\tilde{V}_t^{x,i}, \tilde{E}_t^{x,j})$	A local neighborhood clique w.r.t. i-th landmark object in the measurement graph $\mathcal{G}_t^x$ at time t.
$\mathcal{G}_{t,\text{match}}^x$, $\mathcal{G}_{t,\text{match}}^y$	Two sub-graphs of $\mathcal{G}_t^x$ and $\mathcal{G}_t^y$ respectively, containing the matched landmark objects.
$s_{t,\text{match}}^{x,i}$, $s_{t,\text{match}}^{y,i}$	The i-th ($i \in \mathbb{N}^+$) pair of matched landmark patches in $\mathcal{G}_{t,\text{match}}^x$ and $\mathcal{G}_{t,\text{match}}^y$ respectively.
$p_{s_{t,\text{match}}^{x,i}}$, $p_{s_{t,\text{match}}^{y,i}}$	The matched points clouds reflected from the landmark patch $s_{t,\text{match}}^{x,i}$ and $s_{t,\text{match}}^{y,i}$ respectively.
$d_{t,\text{match}}^{x,i} = (s_{t,\text{match}}^{x,i}, p_{s_{t,\text{match}}^{x,i}})$, $d_{t,\text{match}}^{y,i} = (s_{t,\text{match}}^{y,i}, p_{s_{t,\text{match}}^{y,i}})$	The i-th ($i \in \mathbb{N}^+$) pair of matched landmark objects in $\mathcal{G}_{t,\text{match}}^x$ and $\mathcal{G}_{t,\text{match}}^y$ respectively.
$R^x(\cdot)$,$R^y(\cdot)$	Resnet.
$P(\cdot)$,$P^x(\cdot)$, $P^y(\cdot)$	PointNet.
$f(\cdot)$, $f^x(\cdot)$, $f^y(\cdot)$	Function used to extract features for landmark objects.
$g(\cdot)$, $g^x(\cdot)$, $g^y(\cdot)$	GAT used to extract features for the graph structured data.

as our world coordinate system. The targeted roads are separated into M partitions, and each road partition has a reference point as the origin together with a local coordinate system (see Fig. 1). We call it the *localization coordinate system*. Each partition has length $\leq L$ and rotation $\leq r$ when the vehicle transverses the partition along the road. We call L the partition length and r the partition rotation. Since the vehicle's location w.r.t. the local coordinate system in one road partition can be safely converted to the location w.r.t. the world coordinate system, in our framework the estimated vehicle's location is w.r.t. the localization coordinate system in each road partition m, $m = 1, 2, \cdots, M$. Besides those two external coordinate systems, the vehicle itself has a coordinate system which is named as the *vehicle coordinate system*. Sensors equipped in the vehicle are referred to this coordinate system. For example, the projection of the LiDAR onto the image plane will use this coordinate system.

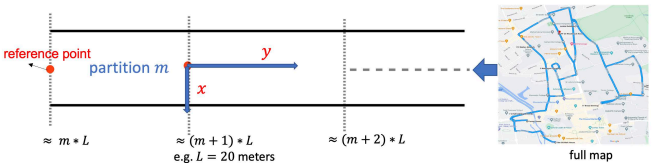

Fig. 1. The localization coordinate system.

B. Localization Pipeline

1) Sensor Data Preprocessing : Similar to [5], we try to extract landmark objects from the environment which will be served as vertices in the graph topological landmark map introduced later in Section III-B.2. To this end, the semantic segmentation network DeepLabV3+ [13] is used to get semantic segmentation outputs for the camera image frames, from which cropped instance patches are obtained. For an illustration, the cropped instance patches shown in Figs. 2c and 2d are examples gotten based on the semantic segmentation shown in Fig. 2b. The static roadside objects, including traffic lights, traffic signs, and poles, are utilized in this paper, while other temporarily static objects like parking cars, are not appropriate landmarks or do not have sufficient distinctive features. The use of building facade feature is being considered in a future work due to the added complexity of appropriately dealing with different objects like windows and balconies on a building facade. We build the 3D LiDAR reference point cloud from all the LiDAR scans captured in the road patition m, similar to the operations in [23]. Additionally we have applied DBSCAN [24] to remove LiDAR point noise. We project the surrounding LiDAR points onto the image frame plane using the intrinsic camera matrix and extrinsic camera matrix. In this paper, for simplicity, only the first image frame in road patition m will be used to construct the map.

We call the cropped instance patches the *landmark patches*. Let $s^{m,j}$ ($j \in \mathbb{N}^+$) denote each landmark patch in the road partition m. Let $p_{s^{m,j}}$ denote the points cloud reflected from landmark patch $s^{m,j}$. As well, $p_{s^{m,j}}$ can be obtained from the LiDAR points mapping and the pixel positions where the landmark patches are located in the original images. DBSCAN is further applied to remove LiDAR outliers.

We call the tuple $d^{m,j} = (s^{m,j}, p_{s^{m,j}})$ a *landmark object* in the road partition m. We use $\{d^{m,j}\}_{j=1}^N = \{d^{m,1}, \cdots, d^{m,N}\}$ to denote the set of the N landmark objects in the road partition m, and use $\{d^{m,j}\}_j$ to denote the set of objects with an arbitrary number rather than N.

2) Landmark Map Graph Construction: We construct a graph topological landmark map $\mathcal{G}^m = (V^m, E^m)$ for calibration in each partition m, $m = 1, 2, \cdots, M$, where V^m is the set of vertices and E^m is the set of edges. The vertex set $V^m = \{d^{m,j}\}_j$ contains the static roadside landmark objects in each partition m obtained from Section III-B.1. One vertex in V^m is connected to the other vertices as its neighbours if and only if they are among the k nearest vertices to the vertex. Here, the coordinate of object $d^{m,j}$ in 3D space is computed by taking the average of all LiDAR points in $p_{s^{m,j}}$, and the distance between two objects is

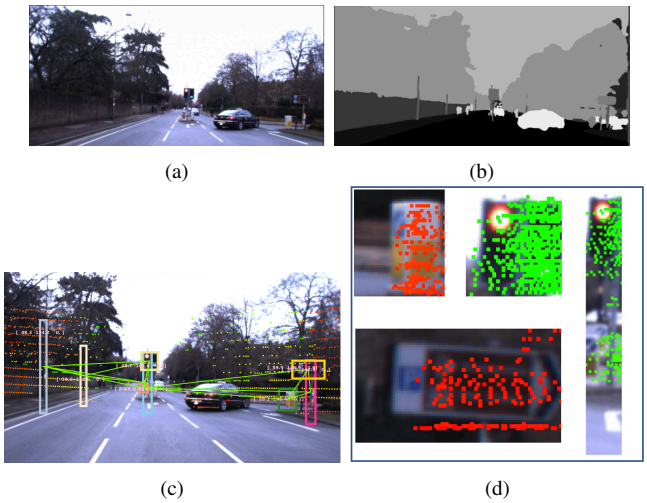

Fig. 2. An example of data preprocessing: (a). Image frame; (b) Semantic segmentation; (c). Green lines indicate the constructed graph edges in our model with landmarks as vertices. (d). Raw landmarks samples.

measured by the L_2 norm. The landmark map shown in Fig. 3 is an example for illustration.

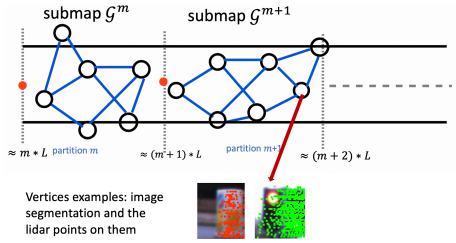

Fig. 3. The graph topological landmark map constructed in each road partition and examples of vertices in the map graph.

3) Real-time Measurement Graph and Localization: At each time t, we utilize the data measured in real-time from vehicle sensors. Specifically, similar to the steps in the landmark map construction, we construct a *measurement graph* $\mathcal{G}_t^x = (V_t^x, E_t^x)$ for time t using the landmark objects extracted similar to Section III-B.2 with the real-time captured camera frames and LiDAR points at time t. A vertex in $\mathcal{G}_t^x$ is connected to the other vertices as its neighbours if and only if they are among the k nearest neighbours to the vertex. We assume the road partition m, where the vehicle is at time t, is known by other rough localization techniques. We may use techniques like EKF, place recognition [23], etc. to do the rough localization when GPS is even not available. We let $\mathcal{G}_t^y$ be $\mathcal{G}_m^y$ if the vehicle at time t is in road partition m.

We next perform matching between the vertices in $\mathcal{G}_t^x = (V_t^x, E_t^x)$ and $\mathcal{G}_t^y = (V_t^y, E_t^y)$ based on GAT network: Here, landmark objects $d_t^{x,i} = (s_t^{x,i}, p_{s_t^{x,i}}) \in V_t^x$ and $d_t^{y,i} = (s_t^{y,i}, p_{s_t^{y,i}}) \in V_t^y$ are said to be *matched* if $s_t^{x,i}$ and $s_t^{y,i}$ are patches of the same static roadside object. The reason for getting the matched landmark objects will be clear in Section III-C and Section III-D. Briefly speaking, our localization neural network (LNN) will compare the features extracted from the matched landmark object pairs in $\mathcal{G}_t^x$ and $\mathcal{G}_t^y$ to do the localization regression. Vertices in $\mathcal{G}_t^x$ and $\mathcal{G}_t^y$ may contains different objects. The representation $d_t^{x,i}$ and $d_t^{y,j}$ may be different even if they are from the same static roadside object, since the data may be collected from different viewpoints with different road conditions. We design a landmark objects matching neural network (LOMNN) to perform the matching between the vertices in $\mathcal{G}_t^x$ and $\mathcal{G}_t^y$. The network is detailed in Section III-C. When the matched pairs of landmark objects are found, the matched landmark objects in two different graphs are determined. We construct two sub-graphs in Section III-D containing the matched landmark object pairs, and denote them as $\mathcal{G}_{t,,\text{match}}^x$ and $\mathcal{G}_{t,\text{match}}^y$ respectively. Different from the conventional matching methods for image patches, our matching method in Section III-C can not only exploit the images of landmark patches but also make full use of the LiDAR points on these patches. We next brief the two network modules.

C. Landmark Objects Matching Based on Graph Attention Networks

We propose a matching method to reveal whether a given pair of landmark objects can be regarded as the same, where '1" and "0" denote the matched label and the mismatched label respectively. It mainly consists of three kinds of neural networks, namely Resnet [14], GAT [25], and PointNet [15], which are respectively used for image feature extraction, neighborhood graph feature description, as well as feature comparison. The specific procedure is shown in Fig. 4. The details are given as follows.

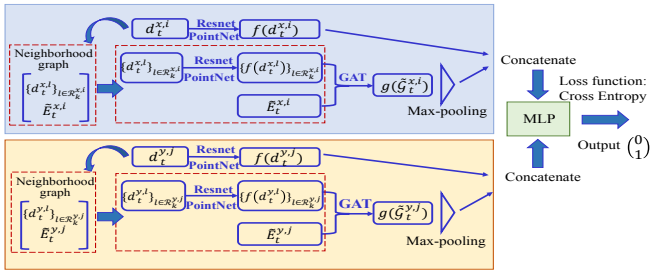

Fig. 4. Matching model based on GAT for landmark objects.

First of all, Resnet and PointNet are used to extract the features of landmark objects $d_t^{x,i}$ and $d_t^{y,j}$, whose outputs are concatenated to obtain the feature representation $f(d_t^{x,i})$ and $f(d_t^{y,j})$. We refer the reader to Section III-D for more details about the module f. Then, for i-th landmark object in $\mathcal{G}_t^x$, we get its k nearest vertices , where we use $\mathcal{R}_k^{x,i}$ to denote the set of the k nearest vertices indices. Given $\mathcal{R}_k^{x,i}$, for each i, we additionally construct a complete graph or clique denoted by $\tilde{\mathcal{G}}_t^{x,i} = (\tilde{V}_t^{x,i}, \tilde{E}_t^{x,i})$, where $\tilde{V}_t^{x,i}$ is the set of vertices, and $\tilde{E}_t^{x,i}$ is set of edges in the clique. $\tilde{\mathcal{G}}_t^{y,j} = (\tilde{V}_t^{y,j}, \tilde{E}_t^{y,j})$ is defined correspondingly for each vertex j in the landmark map graph $\mathcal{G}_t^y$.

Furthermore, GAT is used to describe neighborhood graph features, whose output is $g(\tilde{\mathcal{G}}_t^{x,i})$. Then, the landmark object features and neighborhood graph features are concatenated as the inputs for the final MLP layer where the loss function

is the cross entropy loss, i.e.

$$L_{ce}(\boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}) = -\sum_i \alpha^{(i)} \log \hat{\alpha}^{(i)} + (1-\alpha^{(i)}) \log(1-\hat{\alpha}^{(i)}), \quad (1)$$

in which $\boldsymbol{\alpha} = (\alpha^{(0)}, \alpha^{(1)})$, $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}^{(0)}, \hat{\alpha}^{(1)})$, and $\alpha^{(i)}$ and $\hat{\alpha}^{(i)}$ denote the true label and predicted label from LOMNN respectively.

D. Localization Neural Network

We construct two sub-graphs from $\mathcal{G}_t^x$ and $\mathcal{G}_t^y$ respectively, containing the matched landmark objects obtained from the matching results. We denote them as $\mathcal{G}_{t,\text{match}}^x = (\{d_{t,\text{match}}^{x,i}\}_i, E_{t,\text{match}}^x)$ and $\mathcal{G}_{t,\text{match}}^y = (\{d_{t,\text{match}}^{y,i}\}_i, E_{t,\text{match}}^y)$, where $d_{t,\text{match}}^{x,i} = (s_{t,\text{match}}^{x,i}, p_{s_{t,\text{match}}^{x,i}})$ and $d_{t,\text{match}}^{y,i} = (s_{t,\text{match}}^{y,i}, p_{s_{t,\text{match}}^{y,i}})$ are the i-th matched landmark object pair, and $E_{t,\text{match}}^x$ and $E_{t,\text{match}}^y$ are the sub-graph edges.

Our LNN first uses $f^x(\cdot)$ and $f^y(\cdot)$ to extract features for landmark object pair $d_{t,\text{match}}^{x,i}$ and $d_{t,\text{match}}^{y,i}$ respectively. f (in Section III-C), $f^x(\cdot)$ and $f^y(\cdot)$ have the same network architecture but the separate trainable parameters, which is depicted in Fig. 5. Firstly, Resnets $R^x(\cdot)$ and $R^y(\cdot)$ are used to extract features from the matched landmark patches. Additionally, PointNets $P^x(\cdot)$ and $P^y(\cdot)$, which are applicable for reasoning about unordered data (e.g. LiDAR points) with arbitrary numbers, extract features from points $p_{s_{t,\text{match}}^{x,i}}$ and $p_{s_{t,\text{match}}^{y,i}}$. The two features are concatenated to form the feature of vertex i, namely $f^x(d_{t,\text{match}}^{x,i}) = R^x(s_{t,\text{match}}^{x,i}) || P^x(p_{s_{t,\text{match}}^{x,i}})$, where $||$ denotes the concatenation operation.

We further use GAT $g^x(\cdot)$ to extract features $g^x(d_{t,\text{match}}^{x,i})$ for each landmark object from the graph structured data with $\{f^x(d_{t,\text{match}}^{x,i})\}_i$ and $E_{t,\text{match}}^x$. The similar operation is performed by $g^y(\cdot)$ on the map for $\{f^y(d_{t,\text{match}}^{y,i})\}_i$ and $E_{t,\text{match}}^y$. Here $g^x(\cdot)$ and $g^y(\cdot)$ also have the same network architecture but the separate parameters. Since PointNet is able to deal with unordered data, we then use another PointNet P to perform a comparison for the extracted features from the matched landmark object pairs. Specifically, $\{g^x(d_{t,\text{match}}^{x,i}) - g^y(d_{t,\text{match}}^{y,i})\}_i$ are input to PointNet P. MLP is finally applied to perform the neural network regression using the feature output from PointNet P. The final output of LNN is the vehicle's location w.r.t. the reference point as shown in Figs. 1 and 3. We take the 3-d pose as the output including the translate in x, y direction and the yaw rotation ϕ.

1) Loss Function: We use the Huber loss as our loss function:

$$L_\delta(h, \hat{h}) = \begin{cases} \frac{1}{2}||h-\hat{h}||_2^2 & \text{if } ||h-\hat{h}||_2 < \delta, \\ \delta||h-\hat{h}||_1 - \frac{1}{2}\delta^2 & \text{otherwise,} \end{cases} \quad (2)$$

where h is the ground truth location in the localization coordinate system and $\hat{h}$ is the model output.

2) Training: During the training of LNN, we intentionally perturb the ground truth matching of objects with small probability < 0.1, which makes neural network more robust to deal with potential mismatching output of LOMNN.

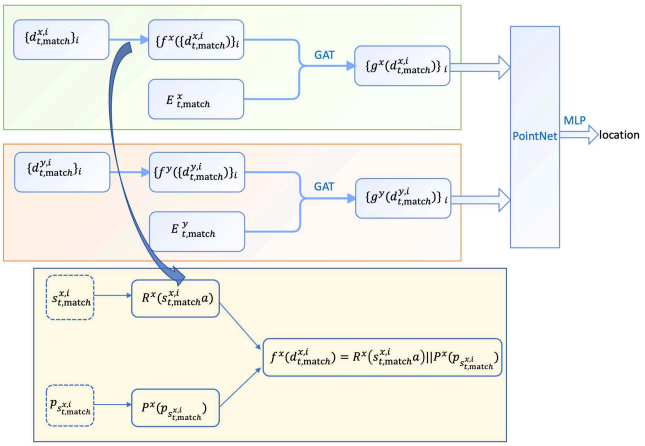

Fig. 5. The overview model for LNN.

IV. Experiment Results

We perform evaluations on two datasets, where one is generated by the CARLA simulator [26] and the other is a real word dataset called Oxford Radar RobotCar Dataset [27]. The Oxford Radar RobotCar Dataset contains camera frames and LiDAR scans captured on the streets in Oxford, UK, by the Point Grey Grasshopper2 (GS2-FW-14S5C-C) Camera and Velodyne HDL-32E Laserscanner, respectively. The resolution of each camera frame in this dataset is 1280 $\times$ 960 pixels. We select two sequences "2019-01-18-14-14-42" and "2019-01-11-12-26-55" from the Oxford Radar RobotCar Dataset. We set the maximum road partition length $L = 20$m (with maximum rotation $r = 30°$). We use sequence "2019-01-18-14-14-42" to perform the landmark map construct using the operations as described in Section III-B.2. Furthermore, for all 29687 image frames in "2019-01-18-14-14-42" we also randomly split them into training and validation sets, with ratio around 3:1. While for sequence "2019-01-11-12-26-55", we use the entire sequence to perform localization test. We compare our localization network LNN with two baseline methods, including DeepLocalization [5] and the simple EKF+GPS algorithm on "2019-01-11-12-26-55". Since not all road partitions have landmark objects, we only compare our LNN with the baselines on the partition that have the requirements to build the landmark map (around 75% partitions satisfying this requirement). For simplicity, in the experiments in this paper, we directly use the rough GPS measurement at time t to determine which road partition m the vehicle is currently in. We use similar settings in the CARLA Dataset, where two runs in the same town are generated. Since CARLA provides us with the landmark object semantic segmentation ground truth, we directly generate landmark patches instead of using DeepLabV3+. The GPS data provided by CARLA are accurate vehicle ground truth location. We therefore only compare our model with DeepLocalization on the Carla Dataset.

A. Performance on Carla Dataset

For the Carla Dataset, we observe from Table II that our method outperforms the other learning-based landmark

approach DeepLocalization. The advantage may come from the explicit landmark matching neural network and the graph map construction approach. DeepLocalization targets at estimating the odometry between two adjacent frames and the integration is performed to get the vehicle real-time location. It may cause large accumulated error even it has used EKF to migrate the risks.

TABLE II
LOCALIZATION PERFORMANCE ON CARLA DATASET.

	x RMSE	y RMSE	yaw ϕ
Ours	0.08m	0.13m	0.35°
DeepLocalization	0.36m	0.45m	0.78°

B. Performance on Oxford Radar RobotCar Dataset

For the Oxford Radar RobotCar Dataset, we observe from Table III that our method outperforms the two baselines. Comparing with Table II, we also observe that on this real dataset, all models perform worse than the the ideal CARLA simulator. This is expected, since the vehicle is in more complex real world scenarios.

TABLE III
LOCALIZATION PERFORMANCE ON OXFORD DATASET.

	x RMSE	y RMSE	yaw ϕ
Ours	0.28m	0.41m	0.71°
DeepLocalization	1.09m	1.35m	1.65°
EKF + GPS	2.19m	2.31m	5.89°

V. CONCLUSION

In autonomous driving applications, it is necessary to investigate high-accuracy vehicular self-localization. We have developed a learning-based localization method based on a calibration landmark map that consists of static roadside objects. We have also studied image patch and LiDAR scan representation for the landmarks as well as the data association task between the real-time captured landmarks and the ones stored in the map. Extensive experiments demonstrate that the static landmarks can bring benefits to AVs' localization accuracy for real-world application. In future work, we will include static building facade [28] elements like windows, door and balcony into our framework to augment the landmark map.

REFERENCES

[1] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments," in *Prof. Conf. Robot. Sci. Syst.*, 2007, p. 1.

[2] J. Wang, "Stochastic modeling for real-time kinematic gps/glonass positioning," *J. Navigation*, vol. 46, no. 4, pp. 297–305, 1999.

[3] D. M. Helmick, Y. Cheng, D. S. Clouse, L. H. Matthies, and S. I. Roumeliotis, "Path following using visual odometry for a mars rover in high-slip environments," in *Proc. IEEE Aerosp.*, 2004, pp. 772–789.

[4] F. Zhang, H. Stähle, G. Chen, C. C. C. Simon, C. Buckl, and A. Knoll, "A sensor fusion approach for localization with cumulative error elimination," in *Proc. Int. Conf. Multisensor Fusion and Integration Intell. Syst*, 2012, pp. 1–6.

[5] N. Engel and etc., "Deeplocalization: Landmark-based self-localization with deep neural networks," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2019, pp. 926–933.

[6] H. Song, W. Choi, and H. Kim, "Robust vision-based relative-localization approach using an rgb-depth camera and lidar sensor fusion," *IEEE Trans. Ind. Electronics*, vol. 63, no. 6, pp. 3725–3736, 2016.

[7] H. Deilamsalehy and T. C. Havens, "Sensor fused three-dimensional localization using imu, camera and lidar," in *Proc. IEEE Sensors*, 2016, pp. 1–3.

[8] L. Huang and M. Barth, "Tightly-coupled lidar and computer vision integration for vehicle detection," in *Prof. IEEE Intell. Vehicles Symp.*, 2009, pp. 604–609.

[9] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 2938–2946.

[10] 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. 2015.

[11] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *Prof. IEEE Int. Conf. Robot Automat.*, 2010, pp. 4372–4378.

[12] C. Brenner, "Vehicle localization using landmarks obtained by a lidar mobile mapping system," *Int. Archives of the Photogrammetry, Remote Sensing and Spatial Inform. Sci.*, vol. 38, no. Part 3A, pp. 139–144, Sept. 2010.

[13] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. vision*, 2018, pp. 801–818.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Int. Conf. Comput. Vision*, 2016, pp. 770–778.

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. IEEE Int. Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 652–660.

[16] L. Jetto, S. Longhi, and G. Venturini, "Development and experimental validation of an adaptive extended kalman filter for the localization of mobile robots," *IEEE Trans. Robot. Automat.*, vol. 15, no. 2, pp. 219–229, Jan. 1999.

[17] R. Van Der Merwe, *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. Oregon Health & Science University, 2004.

[18] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Eur. Conf. Comput. Vision*, 2006, pp. 404–417.

[19] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proc. IEEE Int. Conf. Comput. Vision*, 2011, pp. 2564–2571.

[20] S. Cascianelli, G. Costante, E. Bellocchio, P. Valigi, M. L. Fravolini, and T. A. Ciarfuglia, "Robust visual semi-semantic loop closure detection by a covisibility graph and cnn features," *J. Robot. Auton. Syst.*, vol. 92, pp. 53–65, June 2017.

[21] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *Int. J. Robot. Res.*, vol. 27, no. 6, pp. 647–665, 2008.

[22] Y. Wang, Y. Qiu, P. Cheng, and X. Duan, "Robust loop closure detection integrating visual–spatial–semantic information via topological graphs and cnn features," *J. Remote Sensing*, vol. 12, no. 23, p. 3890, Oct. 2020.

[23] M. A. Uy and G. H. Lee, "Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition," in *Proc. IEEE Int. Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 4470–4479.

[24] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 1–21, 2017.

[25] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.

[26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Annu. Conf. Robot Learn.*, 2017, pp. 1–16.

[27] D. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner, "The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 6433–6438.

[28] W. Ma, W. Ma, S. Xu, and H. Zha, "Pyramid alknet for semantic parsing of building facade image," *IEEE Geosci. Remote Sens. Lett.*, vol. 18, no. 6, pp. 1009–1013, June 2021.