Description

Position determination of a vehicle using image segmentations

Technical Field

The present invention relates to a method for determining a position of a vehicle, to a computer program element for determining a position of a vehicle and to a computer program element for determining matching landmark objects. Further, the invention relates to a computer readable medium.

Background

In autonomous driving, a precisely estimated vehicle position plays a crucial role in the perception, planning, and control functional systems in the autonomous vehicle to perform the driving decisions and actions effectively [1], [2]. The vehicle location is estimated with respect to a global coordinate systems, e.g. Earth-centred Inertial (ECI) coordinate system, Earth-centred Earth-fixed (ECEF) coordinate system and Universal Transverse Mercator (UTM) coordinate systems [3]. In structured or open environments, Global Navigation Satellite Systems (GNSS), such as GPS and GLONASS, usually provide sufficiently accurate localization results. More accurate localization for autonomous vehicles in unreliable GNSS scenarios, e.g. urban or tunnel areas where satellite signals are weakened or blocked, is required. Although more advanced GNSS technology like Differential GPS (DGPS) and Real Time Kinematic (RTK) may mitigate the issue by improving the localization accuracy compared to the conventional GPS, they induce higher cost and still suffer from signal availability due to 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 the autonomous vehicle typically includes Inertial Motion Units (IMU) sensor, wheel odometry sensor, and GPS. The linear accelerations and vehicle angular velocities measured from accelerometers and gyroscopes in IMU, together with the speed and turning measurements from wheel odometry sensors, can be used to estimate the vehicle position relative to its initial position from path integration known as Dead

Reckoning technique [4]. This trivial method has low cost in vehicle localization system, but suffers from accumulated errors even with advanced algorithms [5]. Besides, the magnitude of the localization errors causes inadequacy for autonomous vehicles. Modern vehicle localization systems commonly depend on additional on-board sensors, like cameras, LiDAR, or radar sensors which are also used for other functions of the autonomous vehicle, e.g. Object Detection and Response. Data fusion, the process of combing information from multi-modal sensory data to provide a robust and complete description for the surrounding environment, is therefore required to perform the localization task with less uncertainty and better accuracy results compared to the case when those sources are used individually. In the literature, filtering methods like the basic Kalman filter and its extensions like Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) and Sigma-Point Kalman Filters (SPKF) are commonly used. The recursive estimation process allows the probabilistic descriptions of observations from different sensor models to be included in the Bayes update. Furthermore, quantitative evaluation for the role of each sensor towards the overall performance of the system is possible from the consistent use of statistical measures of uncertainty [6]. In general, complex image data from camera 2 sensors or the point clouds data from LASER scanners are inefficient to be directly utilized in the filtering methods even though they provide rich information. Feature extraction techniques are first employed to extract useful information from the raw data. Traditionally, features of interest include simple point features such as corners, edges and blobs from methods like Scale Invariant Feature Transform (SIFT) [7], Speeded Up Robust Features (SURF)[8], Features From Accelerated Segment Test (FAST) [9] and Oriented Fast and Rotated Brief (ORB) [10]. In 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 [11], [12]. In localization, a reference global or local map need to be first defined. The reference landmark map stores the detected landmarks during the data collection phase and can be categorized into two types: the grid maps [15], [1], [16], [17] and the topological maps[18]. A grid map is presented by an array of structured cells (e.g. the pixels in an image) where each cell represents a region of the environment and contains features of the landmarks therein. In contrast, in a topological map, the landmarks together with the extracted feature

servers as nodes in a graph and the adjacency information reflects the geometric relations between each node. In [1], a detailed grid map is first acquired using a vehicle equipped inertial navigation system and multiple laser range finders, and then a vehicle-mounted LiDAR sensor is used to localize the vehicle relative to the obtained map using a particle filter. In this work, the map is a fixed representation of the environment at each time instance. To achieve robust localization, in [16], maps are regarded as probability distributions over environments in each cell. The accuracy of these methods exceeds that of GPS-based method by over an order of magnitude. In [17], the authors use a LiDAR to obtain a pre-mapped environment, from which landmarks are extracted. When a vehicle transverses the environment, landmarks extracted from vehicle-mounted sensors are used to associate with the LiDAR landmarks and further obtain the vehicle locations. LiDAR is widely used in many other works and show great capability in measuring the ranges of targets in the environment. However, it is weak in recognizing targets [19], which is one of the strong points of computer vision. Thus, in many other works, cameras are adopted to localize vehicles. In [20], [11], 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 performances are influenced by light and weather [19]. Hence, in [21], both LiDAR and RGB-Depth camera are used to do localization by incorporating visual tracking algorithms, depth information of the structured light sensor, and a low-level vision-LiDAR fusion algorithm. In [22], IMU, camera and LiDAR are fused to realize three-dimensional localization. In some works like [12], the measurements need not to be explicitly associated with the landmarks stored in the map. However, in many other works, data association (matching), the process of associating a measurement or feature when a vehicle transverses the environment to its corresponding previously extracted feature, is important. In [1], they 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 measures that use the intensities differences between corresponding pixels in two image patches.

Summary of the invention

There may be a desire to improve position determination for autonomous driving.

The problem is solved by the subject-matter of the independent claims. Embodiments are provided by the dependent claims, the following description and the accompanying figures.

The described embodiments similarly pertain to the method for determining a position of a vehicle, to the computer program element for determining a position of a vehicle, to the computer program element for determining matching landmark objects, and the computer readable medium. Synergetic effects may arise from different combinations of the embodiments although they might not be described in detail.

Further on, it shall be noted that all embodiments of the present invention concerning a method might be carried out with the order of the steps as described, nevertheless this has not to be the only and essential order of the steps of the method. The herein presented methods can be carried out with another order of the disclosed steps without departing from the respective method embodiment, unless explicitly mentioned to the contrary hereinafter.

Technical terms are used by their common sense. If a specific meaning is conveyed to certain terms, definitions of terms will be given in the following in the context of which the terms are used. Moreover, note that in the context of the present invention the terms “position” and “location” are used as synonyms.

According to a first aspect, a method for determining a position of a vehicle is provided. The method comprises the following steps:
In a first step, a reference map is provided. The reference map comprises segmentations of a reference image with landmarks.
In a second step, real time LIDAR data and a measurement image of a vehicle environment are captured.
In a third step, segmentations of the measurement image and neighborhood graphs are determined to obtain a measurement map, wherein a segmentation is

represented by a vertex and where a neighborhood graph comprises the vertex and edges containing information to identify neighboring vertices of the vertex.
In a fourth step, segmentations of the reference image are compared with the segmentations represented by the vertices of the measurement image and the neighborhood graphs and segmentations contained in the reference image and in measurement image are determined.
In a fifth step, the vehicle's location/position is estimated, i.e., determined, with reference to the reference map during its movement along a road based on a result of the comparison.

In the context of the present invention, a segmentation can be regarded as a part of an image. As a simple example, a grid is laid over the image, and a segmentation may be a rectangular grid element in this case. However, the segmentation may be of any shape. In the described method, the segmentations are determined such that they correspond to landmark objects, as described further below in more detail. Although the complete image may be divided into segmentations, segmentations of most interest for the algorithm and method described herein may contain a landmark or a detail of a landmark.

The complete image which is divided into segmentations is also referred to as "frame" herein.

The reference map is determined once in a first phase of the method and is used for calibration purposes. The term "reference image" is also referred to "reference map image". The expression "map" refers in general to a set of graphs containing vertices and their edges". The reference map contains segmentations of static landmarks. Therefore, the reference map is also referred to as landmark map.

As will be explained hereinafter in more detail in the context of particular embodiments, the fifth step, i.e., the estimation of the vehicle's location/position with reference to the reference map during its movement along a road is based on a matching. Said matching will be explained hereinafter in more detail.

According to an embodiment, the step comparing segmentations of the reference image with the segmentations represented by the vertices of the measurement image and the neighborhood graphs comprises the step:
Performing a rough localization to determine a road partition, which the vehicle is currently in, and selecting a reference image from the reference map that is related to the road partition.

A rough position may be used for determining a reference image of the reference map such that the content of the images is similar, that is, contains at least in parts the same object. The rough position may be estimated using an external system such as a navigation system. The navigation system may be a satellite system such as a GNSS system, which may be augmented by an overlay system such as EGNOS or DGNSS (differential GNSS), and/or which may be supported by additional on-board sensors. As is understood by the skilled reader, also other systems can be used.

The expression “rough localization” means that the accuracy of the localization should be sufficient to determine the road partition in which the vehicle is driving at the time of measurement. The accuracy may, for example be half the length interval of the road partition, that is, for example, in the order of half a meter, a meter or even several meters. At the border of a road partition, it is allowed that the determined road partition is falsely a next road partition.

The vehicle’s location is hence estimated from a regression neural network by comparing the extracted features from the equipped sensors’ real-time measurements and the calibration landmark map. By performing the matching between the vertices in the graph of the measurement map and the landmark map common objects are determined, that is snapshots of the same static roadside object.

According to an embodiment, the step comparing segmentations of the reference map with the segmentations represented by the vertices of the measurement image and the neighborhood graphs comprises selecting an object from a set of objects contained in an image, wherein a segmentation represents an object.

In this embodiment, the images contain a set of objects. The segmentations are defined such that they contain an object. The objects for the comparison may be selected randomly. The comparison including the selection are therefore performed repeatedly.

According to an embodiment, the step comparing segmentations of the reference image with the segmentations represented by the vertices of the measurement image and the neighborhood graphs comprises extracting features from a segmentation of a selected object by a first Resnet, and extracting features from the LiDAR data of the selected object by a first PointNet and providing the extracted features to a common PointNet.

As is understood by the skilled reader, a PointNet is a simple and effective Neural Net for point cloud recognition, and a ResNet is known as residual neural network, which is a deep learning model.

In particular, similar to the conventional image processing methods based on neural networks, the Resnets are adopted to extract the high dimensional features for the segmentations. The PointNets are exploited to handle the corresponding LiDAR data, that is, the LiDAR points cloud.

According to an embodiment, the step of comparing segmentations of the reference map with the segmentations represented by the vertices of the measurement image and the neighborhood graphs comprises additionally: extracting features from neighbor segmentations of the segmentation of the selected object and providing the extracted features to a GAT, extracting features from LiDAR points cloud of a neighboring object of the selected object and providing the extracted features to the GAT, and describing the extracted features containing spatial information, and providing the described reference image features to the common PointNet.

This correspond to the previous embodiment, however using neighborhood graphs, that is, the vertices neighbored to the vertex or segmentation of the selected object.

According to an embodiment, the previous steps are performed for the reference image and the measurement image, and the method further comprises the step concatenating the extracted features from the first ResNet, the first PointNet and the GAT and determining a similarity between the features of the object of the reference image and the features of the object of the measurement image.

In other words, all previously described outputs, i.e. from the first Resnet and first PointNet, and of the GAT are provided to the common PoinNet, which actually performs the comparison.

According to an embodiment, the step determining a similarity between the features of the object of the landmark map network part and the features of the object of the measurement map part comprises predicting labels by a Multi-Layer Perceptron (MLP) and calculating a loss function by calculating a cross entropy.

The cross entropy contains the true label and predicted label.

According to an embodiment, the step providing a reference map, the reference map comprising segmentations with landmarks comprises the sub-steps:
capturing LIDAR data points and a reference map image along a road for a road partition,
mapping the LIDAR data points to the reference image,
determining objects on the reference image and landmark segmentations from the reference image using a semantic segmentation neural network, and
constructing a graph topological landmark map containing vertices corresponding each to a segmentation and edges, where an edge identifies neighboring vertices of an vertex.

Images are captured in pre-defined length intervals along the road. These intervals in terms of distance a referred to road partitions. The reference map image

contains at least this part of the road and the environment visible from the point where the image is captured, i.e. the beginning of the interval.

For the determination of objects, a semantic segmentation neural network is used. By the semantic segmentation an intermediated image is obtained that distinguishes object types such as buildings, roads, trees, traffic lights, etc., where details and colors, shades etc. are disregarded, so that it contains semantic information. This intermediate image is used to determine objects which are mapped to segmentations. That is, the segmentations are cropped from the original image using the semantic information of the intermediate image. Only static roadside objects are regarded while dynamic objects are omitted. The resulting segmentation are also referred to as (cropped) instance segmentations or landmark segmentations. The cropped segmentations may be resized to, for example, 256 × 256 pixels with paddings.

A vertex corresponds to a segmentation, and a segmentation corresponds to an object. The object has a coordinate that is determined by averaging over the captured LiDAR points.

The semantic segmentation neural network may be based, for example on a so-called DeepLabV3 architecture.

According to a further aspect, a computer program element for determining matching landmark objects is provided that comprises a landmark map network part and measurement map part. Each of the landmark map network part and measurement map part comprises an object selection module configured to select an object from a set of objects contained in an image, a Resnet configured to extract features from a first segmentation of the selected object and providing the extracted features to a common PointNet, a PointNet configured to extract features from a first LiDAR point cloud of the selected object and providing the extracted features to the common PointNet, a second Resnet configured to extract features from neighbor segmentations of the first segmentation and providing the extracted features to a GAT, and a second PointNet configured to extract features from LiDAR points cloud of the neighboring object of the selected object and

providing the extracted features to a GAT. The GAT is configured to describe the extracted features containing spatial information, and to provide the described landmark map features to the common PointNet. The common PointNet configured to concatenate the extracted features from the first ResNet, the first PointNet and the GAT and to determine a similarity between the features of the object of the landmark map network part and the features of the object of the measurement map part.

The Resnets are adopted to extract the high dimensional features for the landmark segmentations

The computer program element defined in this way is also referred to as Landmark objects matching neural network, LOMNN, herein. As is clear to the skilled reader, any module described herein is to be understood as a software module. It is known to a skilled person that any software operation may also be performed by hardware, that is, any a software module may be replaced by a hardware module performing the same functions.

The computer program element may be part of a computer program, but it can also be an entire program by itself. For example, the computer program element may be used to update an already existing computer program to get to the present invention.

According to a further aspect, a computer readable medium on which a computer program element according to the previous claim is stored is provided.

The computer readable medium may be seen as a storage medium, such as for example, a USB stick, a CD, a DVD, a data storage device, a hard disk, or any other medium on which a program element as described above can be stored.

These and other features, aspects and advantages of the present invention will become better understood with reference to the accompanying figures and the following description.

Other variations to the disclosed embodiments can be understood and effected by those skilled in the art in practicing the claimed invention, from the study of the drawings, the disclosure, and the appended claims. In the claims, the word "comprising" does not exclude other elements or steps and the indefinite article "a" or "an" does not exclude a plurality. A single processor or other unit may fulfill the functions of several items or steps recited in the claims. The mere fact that certain measures are recited in mutually different dependent claims does not indicate that a combination of these measures cannot be used to advantage. Any reference signs in the claims should not be construed as limiting the scope of the claims.

Embodiments and description of the Figures

In the following, embodiments and figures are described in detail.

End to End Learned Localization for AVs

High-accuracy vehicle self-localization is essential for path planning and vehicle safety in autonomous driving. Our goal is to determine the autonomous vehicle's location with respect to a reference map composed of landmarks from the environment which are useful for the localization task. The map is generated off-line using static roadside objects such as traffic lights, traffic signs and poles, and these objects are specifically organized in a graph topology for calibration. An autonomous vehicle is equipped with multi-modal sensors e.g. camera, LiDAR and Global Positioning System (GPS). We use deep learning techniques to perform automatic feature extraction from sensors' measurements. Specifically, we use a Convolutional Neural Network (CNN) to extract features from visual images captured by the vehicle's camera and use a Graph Convolutional Network (GCN) to integrate measurements from the other sensors, e.g. LiDAR. The vehicle's location is estimated from a regression neural network by comparing the extracted features from the equipped sensors' real-time measurements and the calibration landmark map. In our experiments, we perform evaluations using 3 datasets. Our approach achieves state-of-the-art accuracy and is robust to environment changing e.g. variable traffic flows.

I. Background

In autonomous driving, a precisely estimated vehicle position plays a crucial role in the perception, planning, and control functional systems in the autonomous vehicle to perform the driving decisions and actions effectively [1], [2]. The vehicle location is estimated with respect to a global coordinate systems, e.g. Earth-centred Inertial (ECI) coordinate system, Earth-centred Earth-fixed (ECEF) coordinate system and Universal Transverse Mercator (UTM) coordinate systems [3]. In structured or open environments, Global Navigation Satellite Systems (GNSS), such as GPS and GLONASS, usually provide sufficiently accurate localization results. More accurate localization for autonomous vehicles in unreliable GNSS scenarios, e.g. urban or tunnel areas where satellite signals are weakened or blocked, is required. Although more advanced GNSS technology like Differential GPS (DGPS) and Real Time Kinematic (RTK) may mitigate the issue by improving the localization accuracy compared to the conventional GPS, they induce higher cost and still suffer from signal availability due to 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 the autonomous vehicle typically includes Inertial Motion Units (IMU) sensor, wheel odometry sensor, and GPS. The linear accelerations and vehicle angular velocities measured from accelerometers and gyroscopes in IMU, together with the speed and turning measurements from wheel odometry sensors, can be used to estimate the vehicle position relative to its initial position from path integration known as Dead Reckoning technique [4]. This trivial method has low cost in vehicle localization system, but suffers from accumulated errors even with advanced algorithms[5]. Besides, the magnitude of the localization errors causes inadequacy for autonomous vehicles.

Modern vehicle localization systems commonly depend on additional on-board sensors, like cameras, LiDAR, or radar sensors which are also used for other functions of the autonomous vehicle, e.g. Object Detection and Response. Data fusion, the process of combing information from multi-modal sensory data to provide a robust and complete description for the surrounding environment, is therefore required to perform the localization task with less uncertainty and better accuracy results compared to the case when those sources are used individually. In the literature, filtering methods like the basic Kalman filter and its extensions like Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) and Sigma-Point Kalman Filters (SPKF) are commonly used. The recursive estimation process allows the probabilistic descriptions of observations from different sensor models to be included in the Bayes update. Furthermore, quantitative evaluation for the role of each sensor towards the overall performance of the system is possible from the consistent use of statistical measures of uncertainty [6]. In general, complex image data from camera

sensors or the point clouds data from LASER scanners are inefficient to be directly utilized in the filtering methods even though they provide rich information. Feature extraction techniques are first employed to extract useful information from the raw data. Traditionally, features of interest include simple point features such as corners, edges and blobs from methods like Scale Invariant Feature Transform (SIFT) [7], Speeded Up Robust Features (SURF)[8], Features From Accelerated Segment Test (FAST) [9] and Oriented Fast and Rotated Brief (ORB) [10]. In 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 [11], [12]. In our projects, we use Resnet [13] to extract features for landmark segmentations instead of the full camera view images. Similar to [12], we use PointNet [14], however in a different manner, to deal with unordered data. Specifically, [12] use PointNet to process the unordered extracted landmarks location list. While, in our case, we will first use PointNet to directly extract feature from the unordered 3D point clouds measured using LiDAR sensor. In the data fusion procedure, we use the GCN to integrate features from different sensors.

In localization, a reference global or local map need to be first defined. The reference landmark map stores the detected landmarks during the data collection phase and can be categorized into two types: the grid maps[15], [1], [16], [17] and the topological maps[18]. A grid map is presented by an array of structured cells (e.g. the pixels in an image) where each cell represents a region of the environment and contains features of the landmarks therein. In contrast, in a topological map, the landmarks together with the extracted feature servers as nodes in a graph and the adjacency information reflects the geometric relations between each node. In [1], a detailed grid map is first acquired using a vehicle equipped inertial navigation system and multiple laser range finders, and then a vehicle-mounted LiDAR sensor is used to localize the vehicle relative to the obtained map using a particle filter. In this work, the map is a fixed representation of the environment at each time instance. To achieve robust localization, in [16], maps are regarded as probability distributions over environments in each cell. The accuracy of these methods exceeds that of GPS-based method by over an order of magnitude. In [17], the authors use a LiDAR to obtain a pre-mapped environment, from which landmarks are extracted. When a vehicle transverses the environment, landmarks extracted from vehicle-mounted sensors are used to associate with the LiDAR landmarks and further obtain the vehicle locations. LiDAR is widely used in many other works and show great capability in measuring the ranges of targets in the environment. However, it is weak in recognizing targets [19], which is one of the strong points of computer vision. Thus, in many other works, cameras are adopted to localize vehicles. In [20], [11], 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 performances are influenced by light and weather [19]. Hence, in [21], both LiDAR and RGB-Depth camera are used to do localization by incorporating visual tracking algorithms, depth information of the structured light sensor, and a low-level vision-LiDAR fusion algorithm. In [22], IMU, camera and LiDAR are fused to realize three-dimensional localization. In our projects, we use a semantic segmentation algorithm to process the camera images to extract useful static landmarks including traffic lights, traffic signs, poles, road edges. We also utilize the LiDAR point cloud reflected from the landmarks.

Given a reference map for calibration, the localization task aims to determine vehicle position using the measurements from on-board sensors. In some works like [12], the measurements need not to be explicitly associated with the landmarks stored in the map. However, in many other works, data association (matching), the process of associating a measurement or feature when a vehicle transverses the environment to its corresponding previously extracted feature, is important. In [1], they 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 measures that use the intensities differences between corresponding pixels in two image patches. In our projects, we perform the data association using a matching neural network based on Graph Attention Networks (GAT). The output of the network is the similarity between two image patches.

Our project provides a localization solution for such scenarios with precise localization results in

centimeter accuracy. Furthermore, our proposed model is robust to changes of the dynamic environment e.g. different weather conditions and variable traffic flows.

A. Related Work

The problem of environment mapping and vehicle localization is relevant to the Simultaneous Localization and Mapping (SLAM) problem. SLAM, which has been studied for decades [23], [24], [25], is a process in which a robot is required to localize itself in an unknown environment and build a map of this environment at the same time without any prior information with the aid of external sensors. In contrast, in localization task, the reference map can be constructed in the offline phase, and the localization needs to be performed by comparing to the offline reference map. However, a lot of mapping construction techniques from SLAM can be utilized in vehicle localization problem. For example, in [1], they use GRAPHSLAM [26], where an optimization problem is proposed and solved, to get a map representation for the road. For simplicity, in this paper we only consider constructing the map by using the static objects from the frame collected at the reference point in each road partition (see Section II-B2 for details). We will improve the map quality in the future by considering integrating objects from all frames collected in each partition using techniques like GRAPHSLAM to get a consistent map representation from different frames.

In traditional filtering-based localization systems, EKF and particle filter are commonly used. By using laser reflectivity from the flat road surface, [1] has proposed a particle filter method to correlate LiDAR real-time measurements to the reference map.

In some fields, e.g. computer vision [27] and time series processing [28], deep learning approaches using end to end learning show a substantial increase in performance and robustness and a structured and efficient development process by automating feature engineering. Recently, researchers have been trying to leverage the advantages from deep neural networks in localization tasks. Using a kind of CNN called PoseNet that regresses the camera's 6 degrees of freedom pose relative to the scene from a single RGB image, [11] has proposed the first application of deep learning in localization. In [12], they use PointNet [14] to deal with two dynamic inputs of unordered landmarks and measurement lists to regress the vehicle location. With the goal to regress the global pose and simultaneously estimate the odometry between two frames, the authors in [29] have proposed VLocNet based on residual neural networks that take two consecutive monocular images as input. DeepMapping [30] regresses the sensor's pose using a deep neural network followed by a mapping network that models the structure of a scene using grid maps. Contrary to the above methods, our model explicitly use a semantic segmentation algorithm to process the camera images to extract useful static landmarks. We also propose the Landmark Objects Matching Neural Network (LOMNN) to perform data association.

B. Summary of Our Contributions

We have developed a localization framework based on system-level machine learning techniques using multi-modal sensors. In our framework, we first construct a graph topological landmark map for calibration, and the vehicle's location is then estimated with reference to the map during its movement along the road. We use CNN to extract features from visual images captured by the vehicle's camera and use GCN to integrate measurements of the other sensors, e.g. LiDAR. The vehicle's location is estimated from a regression neural network by comparing the extracted features from the equipped sensors' real-time measurements and the calibration landmark map. The details of the framework are described in Section II-B. We also have gotten some preliminary experimental evaluation results using our framework. Specifically, we have tested our model on artificial datasets generated by CARLA [31] and a real dataset provided by Continental Automotive Pte Ltd. The details of the evaluation results are presented in Section III.

II. FRAMEWORK

In this section, we first introduce some fundamental settings of our vehicle. We then 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.

TABLE I
SUMMARY OF KEY NOTATIONS

Notations	Explanations
L	The partition length for a road.
m	The index of road partition where $m = 1, 2, ..., M$.
$\mathcal{G}^m = (V^m, E^m)$	A graph topological landmark map for calibration in the road partition m, where V^m is the set of vertices and E^m is the set of edges of this map.
$s^{m,j}$	The j-th ($j \in \mathbb{N}^+$) landmark segmentation in the road partition m.
$p_{s^{m,j}}$	The points cloud reflected from the landmark segmentation $s^{m,j}$.
$d^{m,j} = (s^{m,j}, p_{s^{m,j}})$	The j-th landmark object in the road partition m.
$\mathcal{G}_t^x = (V_t^x, E_t^x)$	A measurement map based on the x-th frame collected at the time step t, where V_t^x is the set of vertices and E_t^x is the set of edges of this map.
$s_t^{x,i}$	The i-th ($i \in \mathbb{N}^+$) landmark segmentation in the x-th frame collected at the time step t.
$p_{s_t^{x,i}}$	The points cloud reflected from the landmark segmentation $s_t^{x,i}$.
$d_t^{x,i} = (s_t^{x,i}, p_{s_t^{x,i}})$	The i-th landmark object in the x-th frame collected at the time step t.
$\tilde{\mathcal{G}}_t^{x,i} = (\tilde{V}_t^{x,i}, \tilde{E}_t^{x,j})$	A neighborhood map with respect to i-th landmark segmentation in the measurement map $\mathcal{G}_t^{x,i}$ of the x-th frame collected at the time step t, where $\tilde{V}_t^{x,i}$ is the set of vertices and $\tilde{E}_t^{x,j}$ is the set of edges of this map.
$\mathcal{G}_{t,\text{match}}^x$, $\mathcal{G}_{t,\text{match}}^y$	Two sub-graphs containing the common landmark objects in the $\mathcal{G}_t^x$ and $\mathcal{G}_t^y$ respectively.
$s_{t,\text{match}}^{x,i}$, $s_{t,\text{match}}^{y,i}$	The i-th ($i \in \mathbb{N}^+$) pair of matched landmark segmentations 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 segmentation $s_{t,\text{match}}^{x,i}$ 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.
$R(\cdot)$	Resnet used to extract features from landmark segmentations.
$P(\cdot)$	PointNet used to extract features from points clouds.
$f(\cdot)$	It is used to concatenate features from landmark segmentations and points clouds, in other words, it is used to extract features for landmark objects.
$g(\cdot)$	GAT used to extract features for the graph structured data with vertices and edges.

A. Coordinate Systems and Sensors

In our localization framework, there exist three coordinate systems: the world coordinate system, the localization coordinate system and the vehicle coordinate system. We show them in Fig. 1. The world coordinate system can be set as an Earth global surface coordinate system like the Universal Transverse Mercator (UTM) or even some self-defined coordinate systems in a city. In this report, we take the UTM 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. 2). We call it the localization coordinate system. Each partition roughly has length L along a road. We call L the partition length. The output of our framework is a position in the (local) localization coordinate system of the relevant road partition m, $m = 1, 2, \cdots, M$, that the ego-vehicle is currently in. Note that it is safe to transform positions from an earth-fixed local coordinate sytem to a position in a world coordinate system. Beside 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 using this coordinate system. Our probe vehicle used for data acquisition is equipped with cameras, radar, and a LiDAR to mimic any autonomous vehicle, like the CUbE [32]. A detailed description of the sensor setup is shown in Figs. 3 and 4.

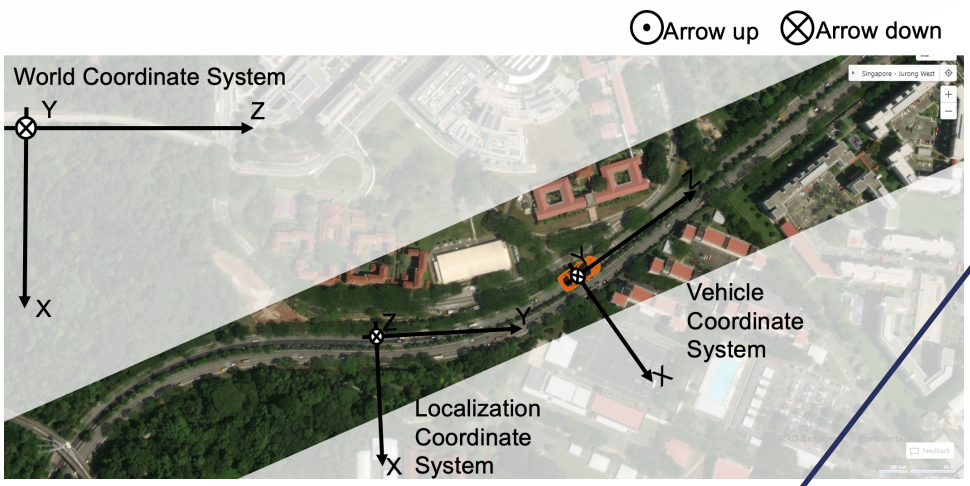


Fig. 1. Coordinate systems.

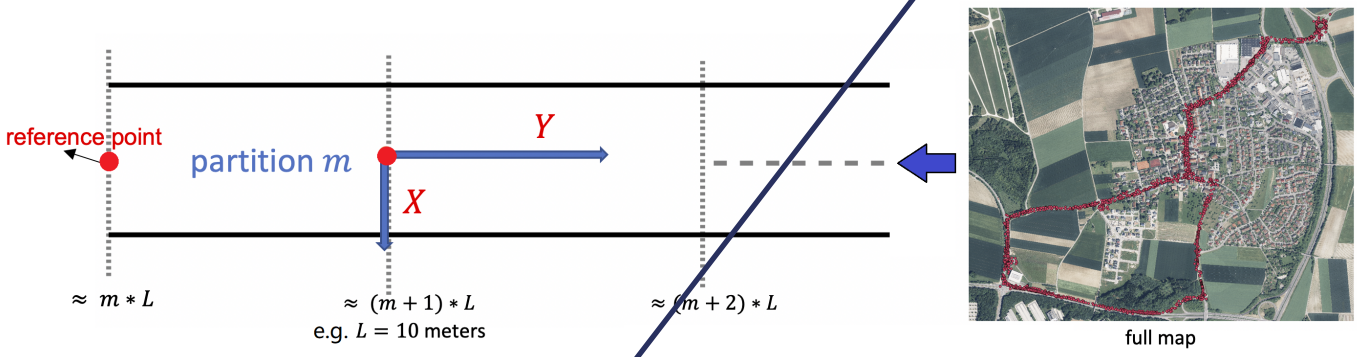


Fig. 2. The localization coordinate system (axis z is leaved out for abbreviation).

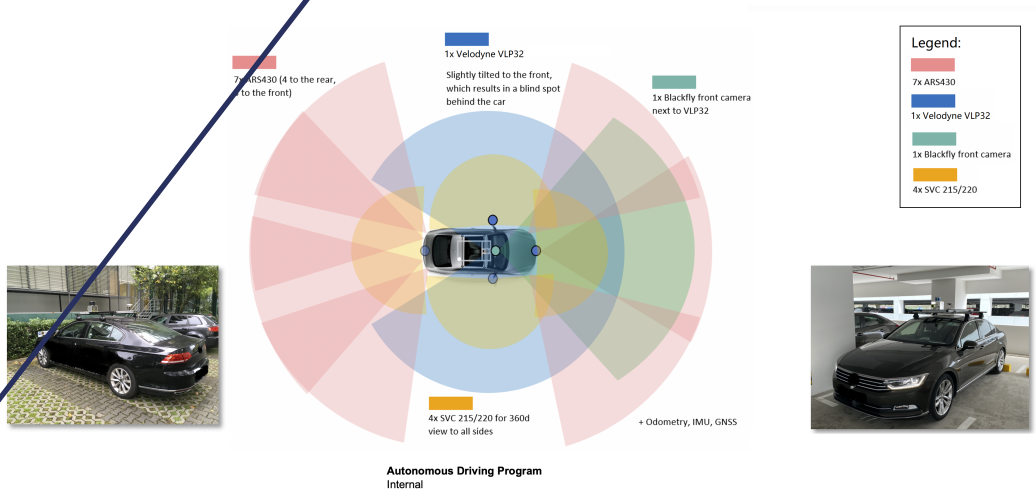


Fig. 3. Sensors equipped in the vehicle.

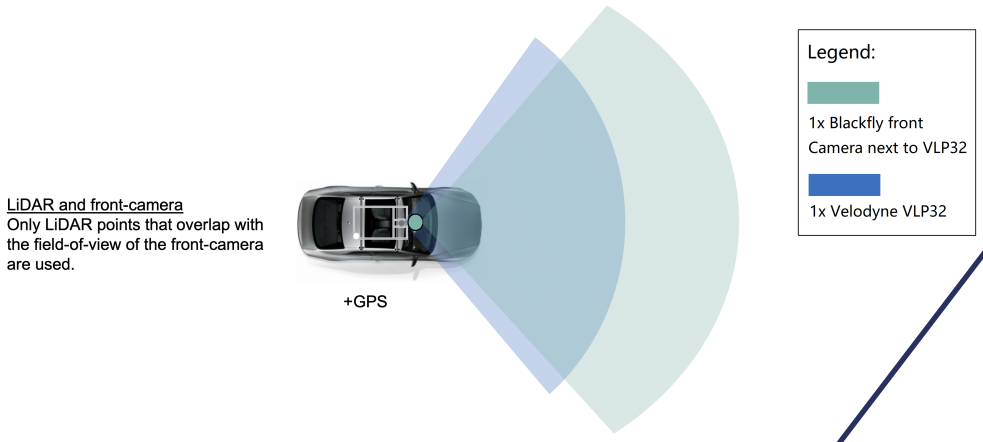


Fig. 4. Sensors used so far.

B. Localization Pipeline

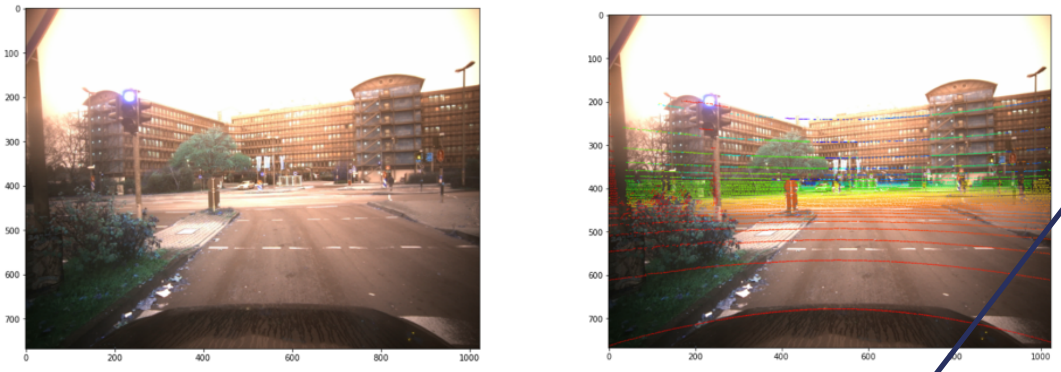
1) Sensor Data Preprocessing : From the sensor data collected in the calibration run on the target roads, we first map LiDAR points collected by Velodyne VLP32 to images captured by the front Blackfly camera using the vehicle coordinate system at each time step. In other words, only LiDAR points that overlap with the field-of-view of the front-camera are used, as Fig. 4 depicts. We use the term *frame* to denote the images and the LiDAR points collected at each time step.

In Fig. 5, we show one example, where Fig. 5a is the original image captured by the front camera at a certain time step and Fig. 5b is the visualization of the LiDAR points mapping. From the images and the mapped LiDAR points, we next try to get objects which will be served as vertices in the graph topological landmark map introduced in Section II-B2. To this end, we use a semantic segmentation neural network based on DeepLabV3 [33] architecture. The model was trained on Cityscapes [34] dataset, and cropped instance segmentations are obtained from it. The cropped instance segmentations shown in Figs. 6b to 6d are examples gotten from the original image Fig. 5a and the semantic segmentation Fig. 6a. We resize all the cropped segmentations to size 256×256 with paddings. The static roadside objects, including traffic lights, traffic signs, poles, road edges, and building windows, are utilized in our localization framework, while other dynamic objects are discarded since they are expected to be anywhere in the second run on the target roads.

We call the cropped instance segmentations the *landmark segmentations* if they are the static roadside object snapshots. Let $s^{m,j}$ ($j \in \mathbb{N}^+$) denote each landmark segmentation in the road partition m. Let $p_{s^{m,j}}$ denote the points cloud reflected from landmark segmentation $s^{m,j}$. As well, $p_{s^{m,j}}$ can be obtained from the LiDAR points mapping and the pixel positions where the landmark segmentations are located in the original images.

We call the tuple $d^{m,j} = (s^{m,j}, p_{s^{m,j}})$ a *landmark object* in the road partition m, if $s^{m,j}$ is a landmark segmentation of the road partition. 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.

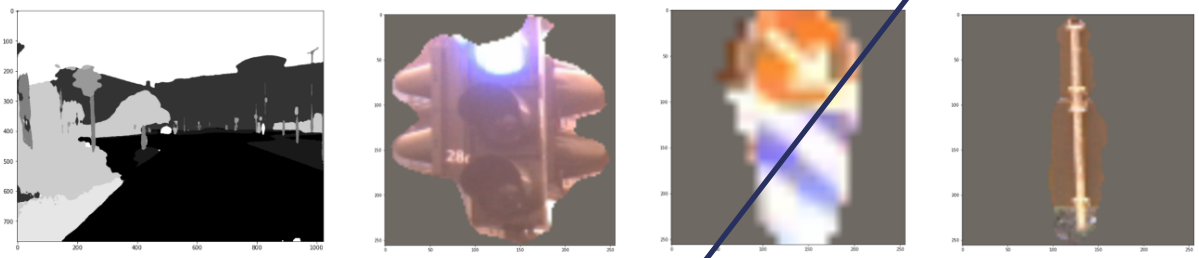
2) Landmark Map Construction: We construct a graph topological landmark map $\mathcal{G}^m = (V^m, E^m)$ for calibration in each partition m, $m = 1, 2, \cdots, M$, where V^m is the set of vertices and E^m is the set of edges. The vertices set $V^m = \{d^{m,j}\}_j$ contains the static roadside landmark objects in each partition m obtained from Section II-B2. 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 $\frac{\sum_{q \in p_{s^{m,j}}} q}{|p_{s^{m,j}}|}$ where $|p_{s^{m,j}}|$ is the the cardinality of $p_{s^{m,j}}$, and the distance between two objects is measured by the L_2 norm. The landmark map shown in



(a) Original image captured by the front camera.

(b) Mapping of the LiDAR points.

Fig. 5. LiDAR points mapping using vehicle coordinate system.



(a) (b) (c) (d)

Fig. 6. A semantic segmentation example (a) from Fig. 5a and landmark segmentations examples (b-d) obtained from (a) and Fig. 5a.

Fig. 7 is an example for illustration. For simplicity, in this report we only consider constructing the map by using the static objects from the frame collected at the reference point in each road partition. We will improve the map quality in the future by considering integrating objects from all frames collected in each partition. In this regard, we denote $G = \{\mathcal{G}^m\}_{m=1}^{M}$.

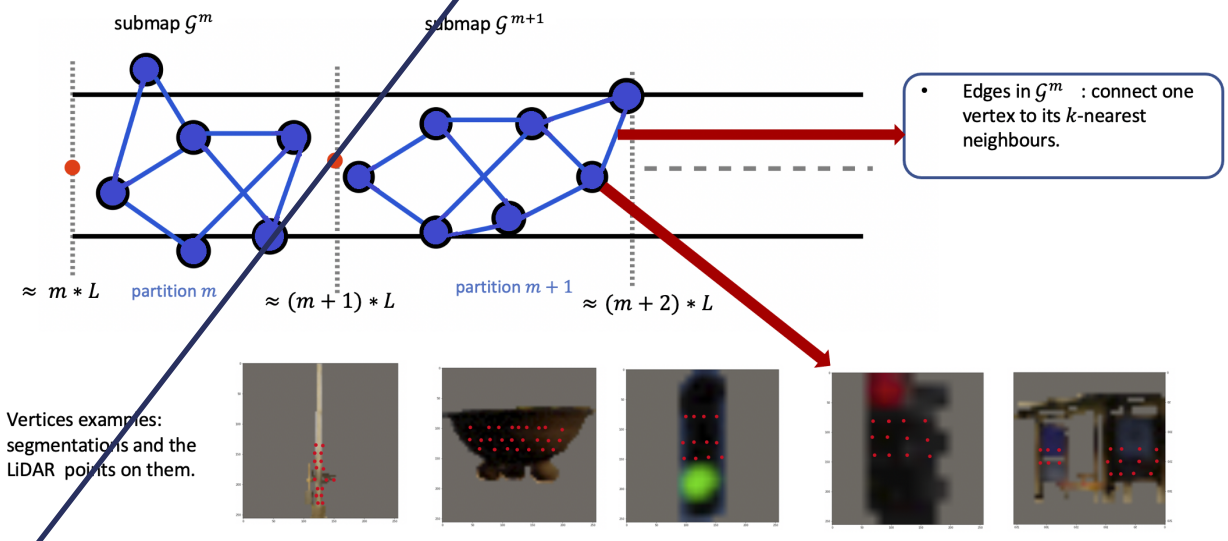


Fig. 7. The graph topological landmark map constructed in each road partition and examples of vertices in the map graph.

3) Real-time Localization:

(i) Get real-time measurements from sensors and construct the measurement graph:

At each time t, we utilize the data measured in real-time from vehicle sensors. Specifically, similar to the steps taken in the landmark map construction, we construct a measurement map $\mathcal{G}_t^x = (V_t^x, E_t^x)$ based on the frame collected at the time step t using the static roadside objects gotten from the image processed by the semantic segmentation neural network and the LiDAR points. 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 vertices to the vertex.

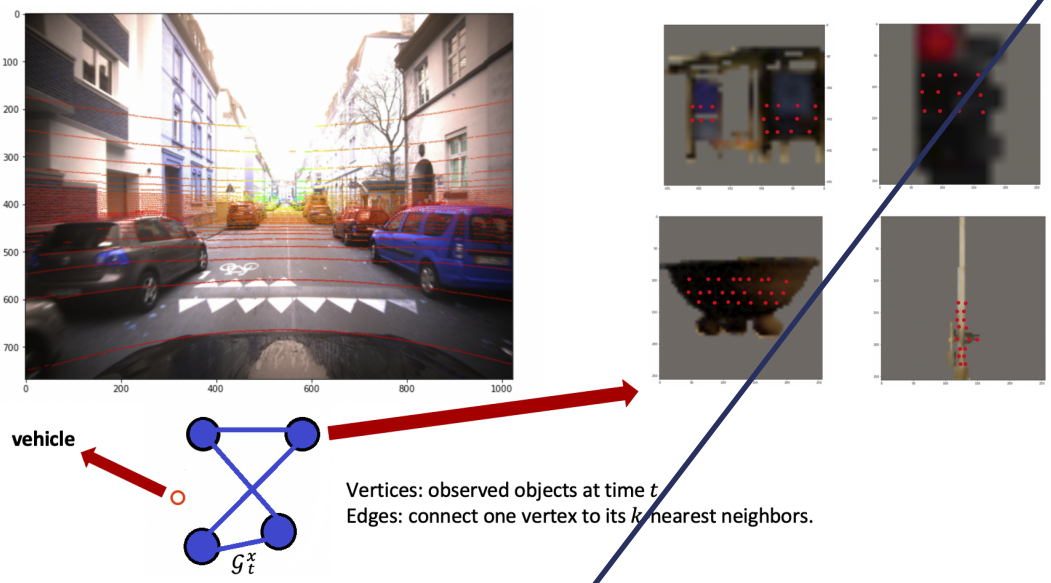


Fig. 8. The graph topological measurement map $\mathcal{G}_t^x$ constructed at time t and examples of vertices in the measurement graph.

(ii) Perform a rough localization to determine which $\mathcal{G}_t^y \in G$ the vehicle is currently in:
For simplicity, we use GPS measured at each time step t to determine the rough location of the vehicle. We may use other techniques like EKF, image matching, etc. to do a rough localization when GPS is even not available. For a concrete illustration, see Fig. 9 and note that if the rough location from GPS at the time step t shows that the vehicle is in road partition $m+1$, we will have $\mathcal{G}_t^y = \mathcal{G}^{m+1}$.

(iii) 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 Graph Attention Networks:
We need to find out the common landmark objects in the landmark map graph $\mathcal{G}_t^y$ and the measurement graph $\mathcal{G}_t^x$. 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 common if $s_t^{x,i}$ and $s_t^{y,i}$ are snapshots of the same static roadside object. The reason for getting the common landmark objects will be clear in Item (iv) and Section II-D. Briefly speaking, our localization neural network (LNN) will compare the features extracted from the common landmark objects 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 contain 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 are collected from different locations with different road conditions. We design a landmark objects matching neural network (LOMNN) to perform matching between the vertices in $\mathcal{G}_t^x$ and $\mathcal{G}_t^y$. The network is detailed in Section II-C. When the matched pairs of landmark objects are found, the common landmark objects in two different graphs are dug out. We construct two sub-graphs containing the common landmark objects, 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 images, our matching method can not only exploit the images of landmark segmentations but also make full use of the LiDAR points on these segmentations.

(iv) Perform localization using LNN :

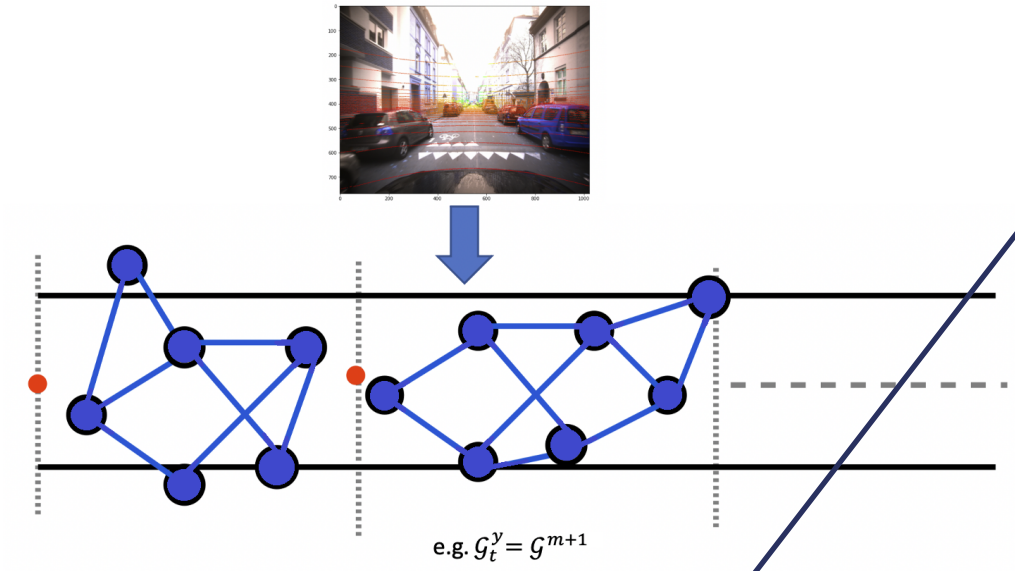


Fig. 9. The rough localization using GPS data.

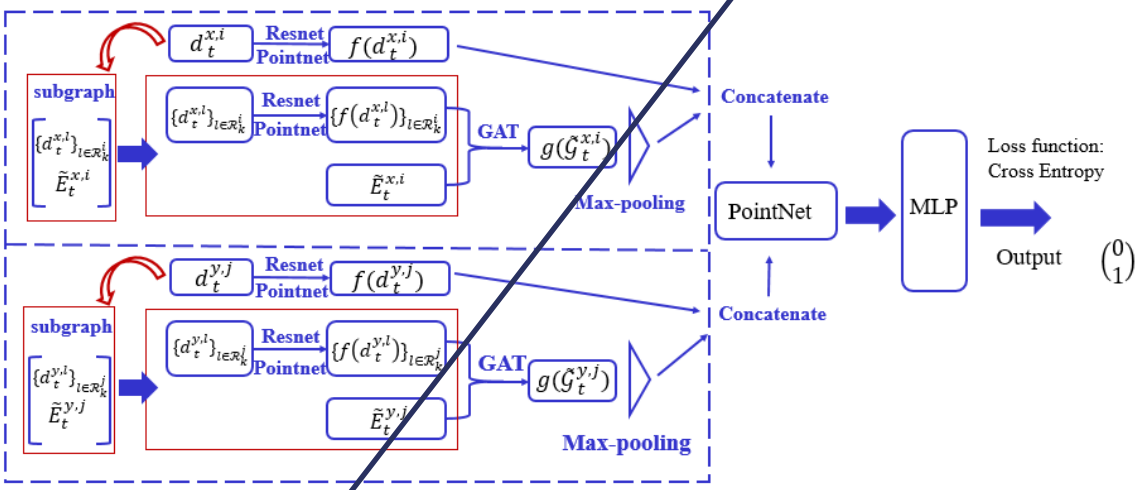


Fig. 10. Matching model based on GAT for landmark segmentations.

LNN can reason about the graph structured data $\mathcal{G}^x_{t,,\text{match}}$ and $\mathcal{G}^y_{t,\text{match}}$ by comparing the real-time measured data to the calibration landmark map. The output of LNN is the estimated vehicle's location with respect to (w.r.t.) the localization coordinate system in each road partition. LNN is introduced in details in Section II-D.

C. Landmark Objects Matching Based on Graph Attention Networks

Here, a matching method is proposed to reveal whether a given pair of landmark objects can be regarded as the same.

1) Matching Model Based on GAT: Let "1" and "0" denote the matched label and the mismatched label, representing whether the pair of landmark objects are matched or not.

As far as the model is concerned, it mainly consists of three kinds of neural networks, namely Resnet [13], GAT [35] and PointNet [14], which are respectively used for feature extraction of images, subgraph feature description, as well as feature comparison. The specific procedure is shown in Fig. 10, whose details are given as follows.

- **Step 1: Feature extraction for landmark objects with Resnet and PointNet.** First of all, two landmark objects $d_t^{x,i} = (s_t^{x,i}, p_{s_t^{x,i}})$ and $d_t^{y,j} = (s_t^{y,j}, p_{s_t^{y,j}})$ are chosen arbitrarily from $\mathcal{G}_t^x$ and $\mathcal{G}_t^y$. Then, the Resnets and PointNets are used to extract the features of $d_t^{x,i}$ and $d_t^{y,j}$. In particular, similar to the conventional image processing methods based on neural networks, the Resnets are adopted to extract the high dimensional features for the landmark segmentations $s_t^{x,i}$ and $s_t^{y,j}$, whose outputs are denoted by $R(s_t^{x,i})$ and $R(s_t^{y,j})$. Meanwhile, the PointNets are exploited to handle the corresponding LiDAR points cloud $p_{s_t^{x,i}}$ and $p_{s_t^{y,j}}$, whose outputs are denoted by $P(p_{s_t^{x,i}})$ and $P(p_{s_t^{y,j}})$. Thus, the extracted concatenated features of $d_t^{x,i}$ and $d_t^{y,j}$ are denoted by $f(d_t^{x,i})$ and $f(d_t^{y,j})$ respectively, where $f(d_t^{x,i}) = (R(s_t^{x,i})||P(p_{s_t^{x,i}}))$ and $f(d_t^{y,j}) = (R(s_t^{y,j})||P(p_{s_t^{y,j}}))$, as well as $||$ denotes the concatenation operation.
- **Step 2: Feature description with GAT.**
First of all, with respect to the landmark objects $d_t^{x,i} = (s_t^{x,i}, p_{s_t^{x,i}})$ and $d_t^{y,j} = (s_t^{y,j}, p_{s_t^{y,j}})$, the corresponding neighborhood maps denoted by $\tilde{\mathcal{G}}_t^{x,i} = (\tilde{V}_t^{x,i}, \tilde{E}_t^{x,i})$ and $\tilde{\mathcal{G}}_t^{y,j} = (\tilde{V}_t^{y,j}, \tilde{E}_t^{y,j})$ are constructed, where $\tilde{V}_t^{x,i}$ and $\tilde{V}_t^{y,j}$ are the set of vertices, as well as $\tilde{E}_t^{x,i}$ and $\tilde{E}_t^{x,i}$ are the set of edges. Specifically, $\tilde{V}_t^{x,i}$ consists of the k nearest landmark objects $\{d_t^{x,l}\}_{l\in\mathcal{R}_k^i}$ to the $d_t^{x,i}$ (which is regarded as the i-th vertex in $\mathcal{G}_t^{x,i}$) and $\tilde{E}_t^{x,i}$ denotes the corresponding connection relationships matrix for these landmark objects. As well, $\mathcal{R}_k^i$ denotes the set of subscripts of k nearest vertices to the i-th vertex in $\mathcal{G}_t^{x,i}$. Similarly, $\tilde{V}_t^{y,j}$ and $\tilde{E}_t^{y,j}$ are the notations like $\tilde{V}_t^{x,i}$ and $\tilde{E}_t^{x,i}$. Then, the landmark objects in $\tilde{\mathcal{G}}_t^{x,i}$ and $\tilde{\mathcal{G}}_t^{y,j}$ are processed by Resnets and PointNets to extract the corresponding features. That is, the landmark segmentations in $\tilde{\mathcal{G}}_t^{x,i}$ and $\tilde{\mathcal{G}}_t^{y,j}$ are processed by means of Resnets and the corresponding LiDAR points cloud are handled by PointNets. Furthermore, the features of landmark objects and the corresponding connection relationship matrices in the $\tilde{\mathcal{G}}_t^{x,i}$ and $\tilde{\mathcal{G}}_t^{y,j}$ are handled by GAT to describe their features containing spatial information. Specifically, taking the graph $\tilde{\mathcal{G}}_t^{x,i}$ as an example [35], the attention mechanism of graph attentional layer w.r.t the pair of vertices (u, v) is described as

$$\xi_{uv} = \frac{\exp(\text{LeakyReLU}(\boldsymbol{a}^T[W \times f(d_t^{x,u})||W \times f(d_t^{x,v})]))}{\sum_{l\in\mathcal{R}_k^i}\exp(\text{LeakyReLU}(\boldsymbol{a}^T[W \times f(d_t^{x,u})||W \times f(d_t^{x,l})]))}, \ (u, v \in \mathcal{R}_k^i), \tag{1}$$

where $(\cdot)^T$ denotes transposition, $||$ is the concatenation operation, ξ_{uv} is the attention coefficient of vertex v to u, W denotes the weight matrix for each vertex, as well as $\boldsymbol{a}$ is the weight vector. It is noted that LeakyReLU nonlinearity is used here. When the multi-head attention is considered, that is, M independent attention mechanisms are used to stabilize the learning process, the output features of each vertex (after applying a nonlinearity σ) are given by

$$f'(d_t^{x,u}) = \Big\|_{r=1}^{M} \sigma\Big(\sum_{v\in\mathcal{R}_k^i} \xi_{uv}^r \times W^r \times f(d_t^{x,v})\Big), \tag{2}$$

especially, the output of the final (prediction) layer is described as

$$f'(d_t^{x,u}) = \sigma\Big(\frac{1}{M}\sum_{r=1}^{M}\sum_{v\in\mathcal{R}_k^i} \xi_{uv}^r \times W^r \times f(d_t^{x,v})\Big). \tag{3}$$

At last, the whole output from GAT, namely $g(\tilde{\mathcal{G}}_t^{x,i})$, consists of all $f'(d_t^{x,u})$ $(u \in \mathcal{R}_k^i)$.
- **Step 3: Comparison based on neural networks.** In this step, outputs in the previous two steps are concatenated as the inputs for the PointNet which is regarded as the similarity measure for the feature comparison. Moreover, the Multi-Layer Perceptron (MLP) is introduced to output the predicted labels, where the loss function is the cross entropy loss, i.e.

$$L_{ce}(\boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}) = -\sum_i \alpha^{(i)} \log \hat{\alpha}^{(i)} + (1 - \alpha^{(i)}) \log(1 - \hat{\alpha}^{(i)}), \tag{4}$$

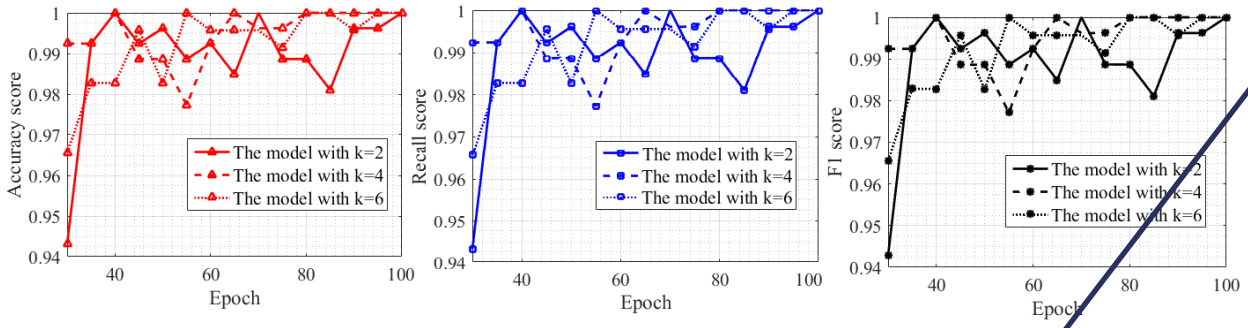


Fig. 11. Experiment results for the matched pairs of landmark segmentations including traffic lights, traffic signs and lamp posts.

in which $\alpha^{(i)}$ and $\hat{\alpha}^{(i)}$ denote the true label and predicted label respectively.

- **Step 4: Matched sub-graph construction.** After finding out the common landmark objects in the landmark map graph $\mathcal{G}_t^y$ and the measurement graph $\mathcal{G}_t^x$, we construct two sub-graphs containing the common landmark objects, and denote them as $\mathcal{G}_{t,,\text{match}}^x$ and $\mathcal{G}_{t,\text{match}}^y$ respectively. Two vertices in $\mathcal{G}_{t,\text{match}}^x$ (respectively, $\mathcal{G}_{t,\text{match}}^y$) are connected to each other if and only if the two vertices are connected to each other in $\mathcal{G}_t^x$ (respectively, $\mathcal{G}_t^y$) .

2) Experiment Results and Discussion: In this subsection, the experiments are taken to evaluate the performance of the proposed matching method. The real data from Continental Automotive Pte Ltd is used to train and test the model.

Particularly, the landmark segmentations include traffic lights, traffic signs and lamp posts, where there are also a number of LiDAR points on them. As well, approximate $5,000$ pairs of segmentations are used, in which 80% pairs with labels are training data and 20% pairs are testing data. In the training process, there are 100 epochs for our model to reveal the corresponding matching performance. As for the criterions, Accuracy score, Recall score and F_1 score are considered to show the matching performance. In addition, different number for k nearest neighbors in neighborhood maps are also chosen in the proposed model so that we can see if number k has great impact on the matching performance.

From Fig. 11, it is illustrated that the proposed model has pretty good performance on landmark objects matching. With different k in this model, the corresponding Accuracy score, Recall score and F_1 score are almost over 98%. This also indicates that there is no too much improvement by using different number for the k nearest neighbors.

D. Localization Neural Network

Our LNN first uses $f^x(\cdot)$ and $f^y(\cdot)$ to extract features for landmark objects $d_{t,\text{match}}^{x,i} = (s_{t,\text{match}}^{x,i}, p_{s_{t,\text{match}}^{x,i}})$ and $d_{t,\text{match}}^{y,i} = (s_{t,\text{match}}^{y,i}, p_{s_{t,\text{match}}^{y,i}})$ in $\mathcal{G}_{t,\text{match}}^x$ and $\mathcal{G}_{t,\text{match}}^y$ respectively. $f^x(\cdot)$ and $f^y(\cdot)$ have the same network architecture but separate trainable parameters, which is depicted in Fig. 12 and Fig. 13. Similar to the operation in Section II-C1, Resnets $R^x(\cdot)$ and $R^y(\cdot)$ are used to extract features from the common landmark segmentations. Additionally, PointNets $P^x(\cdot)$ and $P^y(\cdot)$, which are applicable for reasoning about unordered data (e.g. LiDAR points) with arbitrary numbers (mainly because of the max-pooling operator in PointNet), extract features from $p_{s_{t,\text{match}}^{x,i}}$ and $p_{s_{t,\text{match}}^{y,i}}$. GAT $g^x(\cdot)$ will further 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 $E_{t,\text{match}}^x$, where $f^x(d_{t,\text{match}}^{x,i}) = (R^x(s_{t,\text{match}}^{x,i}) || P^x(p_{s_{t,\text{match}}^{x,i}}))$ is the concatenated features of vertex i, as well as $||$ denotes the concatenation operation. The similar operation is performed by $g^y(\cdot)$ on $\{f^y(d_{t,\text{match}}^{y,i})\}_i$ and $E_{t,\text{match}}^y$. Here $f^x(\cdot)$ and $f^y(\cdot)$ also have the same network architecture but separate parameters. This the same case for $g^x(\cdot)$ and $g^y(\cdot)$. RNN (e.g. LSTM) can be applied here after GAT and will perform learning from measurements history $\{\mathcal{G}_{t'}^x\}_{t'=1}^{t-1}$ during the driving time. However, in the experimental evaluation results

presented in Section III, this RNN structure is left out for simplicity. We will do more tests with the RNN structure in the future. A new PointNet then performs a comparison for the extracted features from the pairs of common landmark objects. MLP is finally applied to perform the neural network regression using the output of the final features from this PointNet.

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

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

where g is the ground truth location and $\hat{g}$ is the model output. During the training of LNN, we intentionally perturb the ground truth matching of objects with small probability (less than 0.1), which makes the test result more promising.

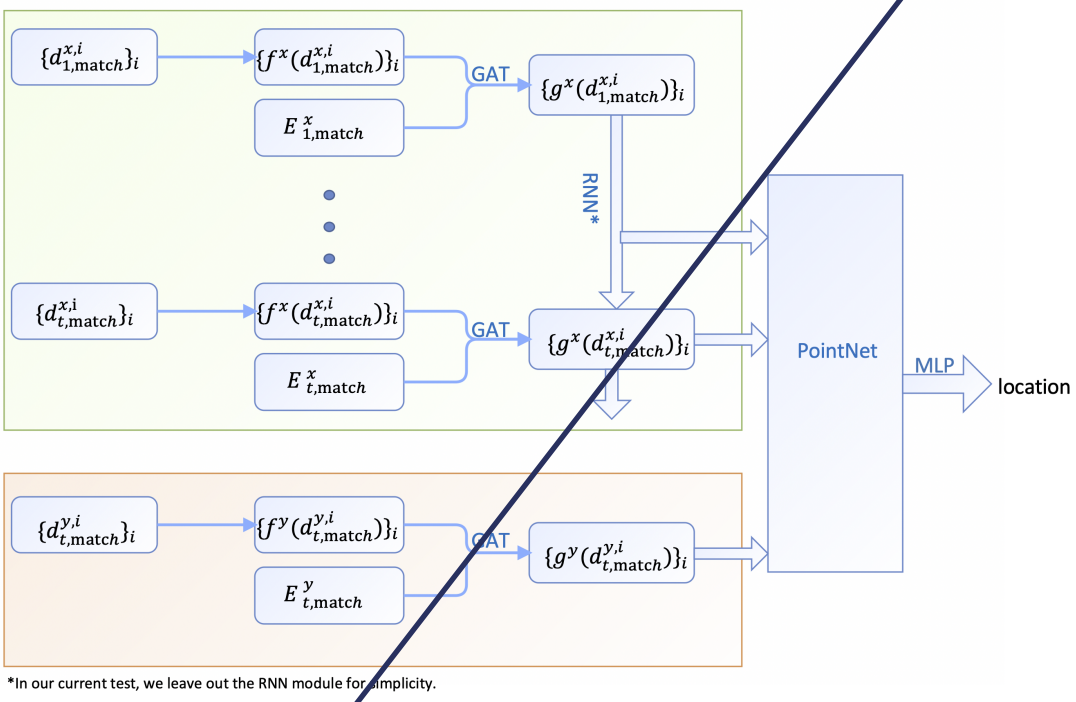


Fig. 12. The overview model for LNN.

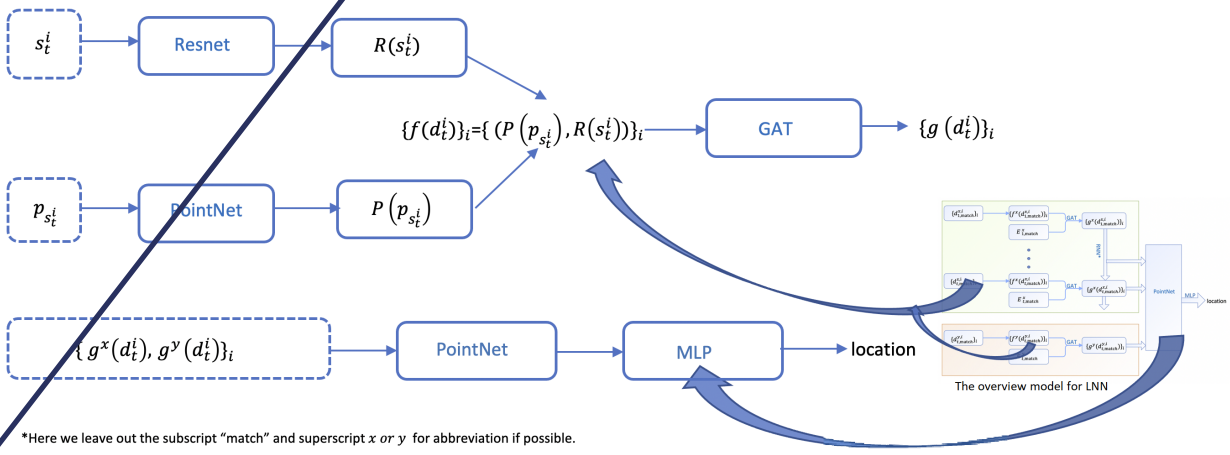


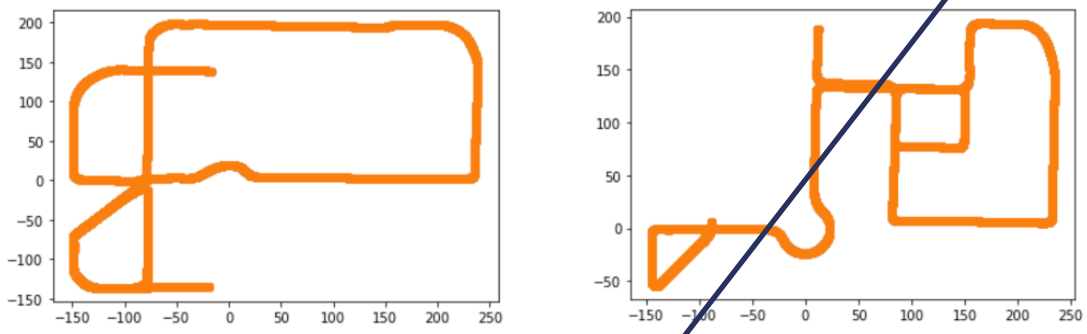
Fig. 13. LNN architecture.

III. Preliminary Test Results

We perform evaluations in two datasets generated by CARLA simulator and a real dataset provided by Continental Automotive Pte Ltd. Instead of the localization results in the real dataset, we get some odometry results using pairs of adjacent frames. This is because in the real dataset the labels are not fully provided. We will do an entire test in the future.

A. Results on Carla Dataset

We use CARLA simulator generate two different datasets with different driving traces in one town, as Fig. 14 depicts, namely CARLA dataset 1 and CARLA dataset 2.



(a) Vehicle movement trace in CARLA dataset 1.

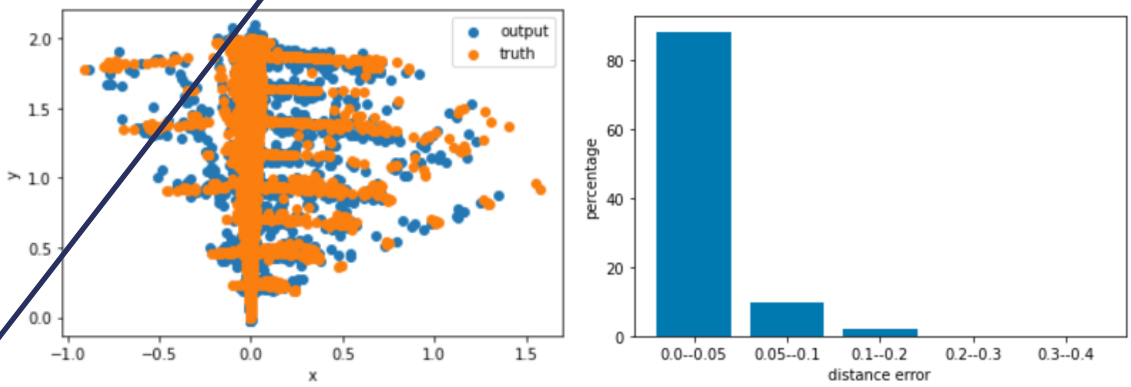
(b) Vehicle movement trace in CARLA dataset 2.

Fig. 14. Vehicle movement trace in different CARLA datasets.

We split the each dataset (around 5000 frames) for training with 80% frames and testing with 20% frames. We include traffic lights, poles, traffic signs, roadside objects (chairs, flowerpots, etc.) as landmark objects and set partition length $L = 2$ (metres). We first train our LNN using the training set in CARLA dataset 1 and test it using the testing set from the same CARLA dataset. The results are shown in Table II and Fig. 15.

TABLE II
Test results in Carla dataset 1 with $L = 2$.

X direction RMSE	Y direction RMSE	Inference time[1]
0.015m	0.066m	0.3s



(a) Output distribution

(b) Error distribution

Fig. 15. Ground truth and model output comparison in CARLA dataset 1 with $L = 2$.

We then use the LNN training in CARLA dataset 1 to perform testing in CARLA dataset 2. The results are shown in Table III. We observe that LNN performs slightly worse with some unseen data from a different driving trace. As well, the centimeter localization accuracy is achieved under the criterion of Root Mean Squared Error (RMSE). There are over 80% prediction results whose distance errors are less than 0.05 metre. Moreover, the mean of inference time is less than 0.5 second in the testing performance, which is acceptable in practice.

TABLE III
TEST RESULTS IN CARLA DATASET 2 WITH $L = 2$.

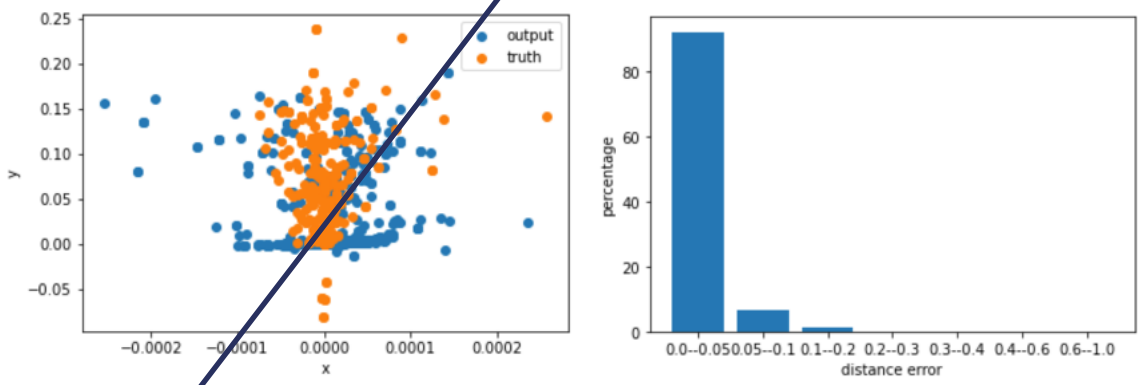
X direction RMSE	Y direction RMSE	Inference time
0.021m	0.078m	$0.3s$

B. Results on Real Dataset

We also test our model in the real dataset provided by Continental Automotive Pte Ltd and get some odometry results using pairs of adjacent frames. The test results are shown in Table IV and Fig. 16. From these, it is apparent that the centimeter localization accuracy is available for the real dataset under the criterion of Root Mean Squared Error (RMSE). There are over 90% prediction results whose distance errors are less than 0.05 metre. Moreover, the mean of inference time in the real dataset is acceptable in practice, which is similar to that in Carla dataset.

TABLE IV
ODOMETRY TEST RESULTS IN THE REAL DATASET.

X direction RMSE	Y direction RMSE	Inference time
0.001m	0.015m	$0.3s$



(a) Output distribution (b) Error distribution

Fig. 16. Ground truth and model output comparison in the real dataset.

REFERENCES

[1] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments," in *Robotics: science and systems*, 2007, p. 1.

[2] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car—part i: Distributed system architecture and development process," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 12, pp. 7131–7140, 2014.

[3] I. Skog and P. Handel, "In-car positioning and navigation technologies—a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 4–21, 2009.

[4] 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 *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No. 04TH8720)*, vol. 2, 2004, pp. 772–789.

[5] 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 *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2012, pp. 1–6.

[6] H. Durrant-Whyte and T. C. Henderson, "Multisensor data fusion," in *Springer handbook of robotics*, 2016, pp. 867–896.

[7] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, 1999, pp. 1150–1157.

[8] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*, 2006, pp. 404–417.

[9] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*, 2006, pp. 430–443.

[10] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*, 2011, pp. 2564–2571.

[11] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[12] N. Engel, S. Hoermann, M. Horn, V. Belagiannis, and K. Dietmayer, "Deeplocalization: Landmark-based self-localization with deep neural networks," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 926–933.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[15] A. Elfes *et al.*, "Occupancy grids: A stochastic spatial representation for active robot perception," in *Proceedings of the Sixth Conference on Uncertainty in AI*, vol. 2929. Morgan Kaufmann, 1990, p. 6.

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

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

[18] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *J. Comput., vol*, vol. 7, no. 3, 1978.

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

[20] H. Lategahn and C. Stiller, "Vision-only localization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pp. 1246–1257, 2014.

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

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

[23] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard, "Efficient estimation of accurate maximum likelihood maps in 3d," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 3472–3478.

[24] M. A. Paskin, "Thin junction tree filters for simultaneous localization and mapping," in *in Int. Joint Conf. on Artificial Intelligence*. Citeseer, 2003.

[25] H. F. Durrant-Whyte, "Uncertain geometry in robotics," *IEEE Journal on Robotics and Automation*, vol. 4, no. 1, pp. 23–31, 1988.

[26] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[28] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[29] A. Valada, N. Radwan, and W. Burgard, "Deep auxiliary learning for visual localization and odometry," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6939–6946.

[30] L. Ding and C. Feng, "Deepmapping: Unsupervised map estimation from multiple point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8650–8659.

[31] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[32] A. Hartmannsgruber, J. Seitz, M. Schreier, M. Strauss, N. Balbierer, and A. Hohm, "Cube: A research platform for shared mobility and autonomous driving in urban environments," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2315–2322.

[33] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[34] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[35] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

Other variations to the disclosed embodiments can be understood and effected by those skilled in the art in practicing the claimed invention, from the study of the drawings, the disclosure, and the appended claims. In the claims, the word "comprising" does not exclude other elements or steps and the indefinite article "a" or "an" does not exclude a plurality. A single processor or other unit may fulfill the functions of several items or steps recited in the claims. The mere fact that certain measures are recited in mutually different dependent claims does not indicate that a combination of these measures cannot be used to advantage. Any reference signs in the claims should not be construed as limiting the scope of the claims.

Embodiments and description of the Figures

In the following, embodiments and figures are described in detail.

Fig. 1. Coordinate systems.

Fig. 2. The localization coordinate system (axis z is leaved out for abbreviation).

Fig. 3. Sensors equipped in the vehicle.

Fig. 4. Sensors used so far.

Fig. 5. LiDAR points mapping using vehicle coordinate system.

Fig. 6. A semantic segmentation example (a) from Fig. 5a and landmark segmentations examples (b-d) obtained from (a) and Fig. 5a.

Fig. 7. The graph topological landmark map constructed in each road partition and examples of vertices in the map graph.

Fig. 8. The graph topological measurement map $\mathcal{G}_t^x$ constructed at time t and examples of vertices in the measurement graph.

Fig. 9. The rough localization using GPS data.

Fig. 10. Matching model based on GAT for landmark segmentations.

Fig. 11. Experiment results for the matched pairs of landmark segmentations including traffic lights, traffic signs and lamp posts.

Fig. 12. The overview model for LNN.

Fig. 13. LNN architecture.

Fig. 14. Vehicle movement trace in different CARLA datasets.

Fig. 15. Ground truth and model output comparison in CARLA dataset 1 with L = 2.

Fig. 16. Ground truth and model output comparison in the real dataset.

12

End to End Learned Localization for AVs

High-accuracy vehicle self-localization is essential for path planning and vehicle safety in autonomous driving. Our goal is to determine the autonomous vehicle's location with respect to a reference map composed of landmarks from the environment which are useful for the localization task. The map is generated off-line using static roadside objects such as traffic lights, traffic signs and poles, and these objects are specifically organized in a graph topology for calibration. An autonomous vehicle is equipped with multi-modal sensors e.g. camera, LiDAR and Global Positioning System (GPS). We use deep learning techniques to perform automatic feature extraction from sensors' measurements. Specifically, we use a Convolutional Neural Network (CNN) to extract features from visual images captured by the vehicle's camera and use a Graph Convolutional Network (GCN) to integrate measurements from the other sensors, e.g. LiDAR. The vehicle's location is estimated from a regression neural network by comparing the extracted features from the equipped sensors' real-time measurements and the calibration landmark map. In our experiments, we perform evaluations using 3 datasets. Our approach achieves state-of-the-art accuracy and is robust to environment changing e.g. variable traffic flows.

I. Background

In autonomous driving, a precisely estimated vehicle position plays a crucial role in the perception, planning, and control functional systems in the autonomous vehicle to perform the driving decisions and actions effectively [1], [2]. The vehicle location is estimated with respect to a global coordinate systems, e.g. Earth-centred Inertial (ECI) coordinate system, Earth-centred Earth-fixed (ECEF) coordinate system and Universal Transverse Mercator (UTM) coordinate systems [3]. In structured or open environments, Global Navigation Satellite Systems (GNSS), such as GPS and GLONASS, usually provide sufficiently accurate localization results. More accurate localization for autonomous vehicles in unreliable GNSS scenarios, e.g. urban or tunnel areas where satellite signals are weakened or blocked, is required. Although more advanced GNSS technology like Differential GPS (DGPS) and Real Time Kinematic (RTK) may mitigate the issue by improving the localization accuracy compared to the conventional GPS, they induce higher cost and still suffer from signal availability due to 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 the autonomous vehicle typically includes Inertial Motion Units (IMU) sensor, wheel odometry sensor, and GPS. The linear accelerations and vehicle angular velocities measured from accelerometers and gyroscopes in IMU, together with the speed and turning measurements from wheel odometry sensors, can be used to estimate the vehicle position relative to its initial position from path integration known as Dead Reckoning technique [4]. This trivial method has low cost in vehicle localization system, but suffers from accumulated errors even with advanced algorithms[5]. Besides, the magnitude of the localization errors causes inadequacy for autonomous vehicles.

Modern vehicle localization systems commonly depend on additional on-board sensors, like cameras, LiDAR, or radar sensors which are also used for other functions of the autonomous vehicle, e.g. Object Detection and Response. Data fusion, the process of combing information from multi-modal sensory data to provide a robust and complete description for the surrounding environment, is therefore required to perform the localization task with less uncertainty and better accuracy results compared to the case when those sources are used individually. In the literature, filtering methods like the basic Kalman filter and its extensions like Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) and Sigma-Point Kalman Filters (SPKF) are commonly used. The recursive estimation process allows the probabilistic descriptions of observations from different sensor models to be included in the Bayes update. Furthermore, quantitative evaluation for the role of each sensor towards the overall performance of the system is possible from the consistent use of statistical measures of uncertainty [6]. In general, complex image data from camera

sensors or the point clouds data from LASER scanners are inefficient to be directly utilized in the filtering methods even though they provide rich information. Feature extraction techniques are first employed to extract useful information from the raw data. Traditionally, features of interest include simple point features such as corners, edges and blobs from methods like Scale Invariant Feature Transform (SIFT) [7], Speeded Up Robust Features (SURF)[8], Features From Accelerated Segment Test (FAST) [9] and Oriented Fast and Rotated Brief (ORB) [10]. In 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 [11], [12]. In our projects, we use Resnet [13] to extract features for landmark segmentations instead of the full camera view images. Similar to [12], we use PointNet [14], however in a different manner, to deal with unordered data. Specifically, [12] use PointNet to process the unordered extracted landmarks location list. While, in our case, we will first use PointNet to directly extract feature from the unordered 3D point clouds measured using LiDAR sensor. In the data fusion procedure, we use the GCN to integrate features from different sensors.

In localization, a reference global or local map need to be first defined. The reference landmark map stores the detected landmarks during the data collection phase and can be categorized into two types: the grid maps[15], [1], [16], [17] and the topological maps[18]. A grid map is presented by an array of structured cells (e.g. the pixels in an image) where each cell represents a region of the environment and contains features of the landmarks therein. In contrast, in a topological map, the landmarks together with the extracted feature servers as nodes in a graph and the adjacency information reflects the geometric relations between each node. In [1], a detailed grid map is first acquired using a vehicle equipped inertial navigation system and multiple laser range finders, and then a vehicle-mounted LiDAR sensor is used to localize the vehicle relative to the obtained map using a particle filter. In this work, the map is a fixed representation of the environment at each time instance. To achieve robust localization, in [16], maps are regarded as probability distributions over environments in each cell. The accuracy of these methods exceeds that of GPS-based method by over an order of magnitude. In [17], the authors use a LiDAR to obtain a pre-mapped environment, from which landmarks are extracted. When a vehicle transverses the environment, landmarks extracted from vehicle-mounted sensors are used to associate with the LiDAR landmarks and further obtain the vehicle locations. LiDAR is widely used in many other works and show great capability in measuring the ranges of targets in the environment. However, it is weak in recognizing targets [19], which is one of the strong points of computer vision. Thus, in many other works, cameras are adopted to localize vehicles. In [20], [11], 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 performances are influenced by light and weather [19]. Hence, in [21], both LiDAR and RGB-Depth camera are used to do localization by incorporating visual tracking algorithms, depth information of the structured light sensor, and a low-level vision-LiDAR fusion algorithm. In [22], IMU, camera and LiDAR are fused to realize three-dimensional localization. In our projects, we use a semantic segmentation algorithm to process the camera images to extract useful static landmarks including traffic lights, traffic signs, poles, road edges. We also utilize the LiDAR point cloud reflected from the landmarks.

Given a reference map for calibration, the localization task aims to determine vehicle position using the measurements from on-board sensors. In some works like [12], the measurements need not to be explicitly associated with the landmarks stored in the map. However, in many other works, data association (matching), the process of associating a measurement or feature when a vehicle transverses the environment to its corresponding previously extracted feature, is important. In [1], they 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 measures that use the intensities differences between corresponding pixels in two image patches. In our projects, we perform the data association using a matching neural network based on Graph Attention Networks (GAT). The output of the network is the similarity between two image patches.

Our project provides a localization solution for such scenarios with precise localization results in

centimeter accuracy. Furthermore, our proposed model is robust to changes of the dynamic environment e.g. different weather conditions and variable traffic flows.

A. Related Work

The problem of environment mapping and vehicle localization is relevant to the Simultaneous Localization and Mapping (SLAM) problem. SLAM, which has been studied for decades [23], [24], [25], is a process in which a robot is required to localize itself in an unknown environment and build a map of this environment at the same time without any prior information with the aid of external sensors. In contrast, in localization task, the reference map can be constructed in the offline phase, and the localization needs to be performed by comparing to the offline reference map. However, a lot of mapping construction techniques from SLAM can be utilized in vehicle localization problem. For example, in [1], they use GRAPHSLAM [26], where an optimization problem is proposed and solved, to get a map representation for the road. For simplicity, in this paper we only consider constructing the map by using the static objects from the frame collected at the reference point in each road partition (see Section II-B2 for details). We will improve the map quality in the future by considering integrating objects from all frames collected in each partition using techniques like GRAPHSLAM to get a consistent map representation from different frames.

In traditional filtering-based localization systems, EKF and particle filter are commonly used. By using laser reflectivity from the flat road surface, [1] has proposed a particle filter method to correlate LiDAR real-time measurements to the reference map.

In some fields, e.g. computer vision [27] and time series processing [28], deep learning approaches using end to end learning show a substantial increase in performance and robustness and a structured and efficient development process by automating feature engineering. Recently, researchers have been trying to leverage the advantages from deep neural networks in localization tasks. Using a kind of CNN called PoseNet that regresses the camera's 6 degrees of freedom pose relative to the scene from a single RGB image, [11] has proposed the first application of deep learning in localization. In [12], they use PointNet [14] to deal with two dynamic inputs of unordered landmarks and measurement lists to regress the vehicle location. With the goal to regress the global pose and simultaneously estimate the odometry between two frames, the authors in [29] have proposed VLocNet based on residual neural networks that take two consecutive monocular images as input. DeepMapping [30] regresses the sensor's pose using a deep neural network followed by a mapping network that models the structure of a scene using grid maps. Contrary to the above methods, our model explicitly use a semantic segmentation algorithm to process the camera images to extract useful static landmarks. We also propose the Landmark Objects Matching Neural Network (LOMNN) to perform data association.

B. Summary of Our Contributions

We have developed a localization framework based on system-level machine learning techniques using multi-modal sensors. In our framework, we first construct a graph topological landmark map for calibration, and the vehicle's location is then estimated with reference to the map during its movement along the road. We use CNN to extract features from visual images captured by the vehicle's camera and use GCN to integrate measurements of the other sensors, e.g. LiDAR. The vehicle's location is estimated from a regression neural network by comparing the extracted features from the equipped sensors' real-time measurements and the calibration landmark map. The details of the framework are described in Section II-B. We also have gotten some preliminary experimental evaluation results using our framework. Specifically, we have tested our model on artificial datasets generated by CARLA [31] and a real dataset provided by Continental Automotive Pte Ltd. The details of the evaluation results are presented in Section III.

II. Framework

In this section, we first introduce some fundamental settings of our vehicle. We then 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.

TABLE I
SUMMARY OF KEY NOTATIONS

Notations	Explanations
L	The partition length for a road.
m	The index of road partition where $m = 1, 2, ..., M$.
$\mathcal{G}^m = (V^m, E^m)$	A graph topological landmark map for calibration in the road partition m, where V^m is the set of vertices and E^m is the set of edges of this map.
$s^{m,j}$	The j-th ($j \in \mathbb{N}^+$) landmark segmentation in the road partition m.
$p_{s^{m,j}}$	The points cloud reflected from the landmark segmentation $s^{m,j}$.
$d^{m,j} = (s^{m,j}, p_{s^{m,j}})$	The j-th landmark object in the road partition m.
$\mathcal{G}_t^x = (V_t^x, E_t^x)$	A measurement map based on the x-th frame collected at the time step t, where V_t^x is the set of vertices and E_t^x is the set of edges of this map.
$s_t^{x,i}$	The i-th ($i \in \mathbb{N}^+$) landmark segmentation in the x-th frame collected at the time step t.
$p_{s_t^{x,i}}$	The points cloud reflected from the landmark segmentation $s_t^{x,i}$.
$d_t^{x,i} = (s_t^{x,i}, p_{s_t^{x,i}})$	The i-th landmark object in the x-th frame collected at the time step t.
$\tilde{\mathcal{G}}_t^{x,i} = (\tilde{V}_t^{x,i}, \tilde{E}_t^{x,j})$	A neighborhood map with respect to i-th landmark segmentation in the measurement map $\mathcal{G}_t^{x,i}$ of the x-th frame collected at the time step t, where $\tilde{V}_t^{x,i}$ is the set of vertices and $\tilde{E}_t^{x,j}$ is the set of edges of this map.
$\mathcal{G}_{t,\text{match}}^x$, $\mathcal{G}_{t,\text{match}}^y$	Two sub-graphs containing the common landmark objects in the $\mathcal{G}_t^x$ and $\mathcal{G}_t^y$ respectively.
$s_{t,\text{match}}^{x,i}$, $s_{t,\text{match}}^{y,i}$	The i-th ($i \in \mathbb{N}^+$) pair of matched landmark segmentations 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 segmentation $s_{t,\text{match}}^{x,i}$ 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.
$R(\cdot)$	Resnet used to extract features from landmark segmentations.
$P(\cdot)$	PointNet used to extract features from points clouds.
$f(\cdot)$	It is used to concatenate features from landmark segmentations and points clouds, in other words, it is used to extract features for landmark objects.
$g(\cdot)$	GAT used to extract features for the graph structured data with vertices and edges.

A. Coordinate Systems and Sensors

In our localization framework, there exist three coordinate systems: the world coordinate system, the localization coordinate system and the vehicle coordinate system. We show them in Fig. 1. The world coordinate system can be set as an Earth global surface coordinate system like the Universal Transverse Mercator (UTM) or even some self-defined coordinate systems in a city. In this report, we take the UTM 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. 2). We call it the localization coordinate system. Each partition roughly has length L along a road. We call L the partition length. The output of our framework is a position in the (local) localization coordinate system of the relevant road partition m, $m = 1, 2, \cdots, M$, that the ego-vehicle is currently in. Note that it is safe to transform positions from an earth-fixed local coordinate sytem to a position in a world coordinate system. Beside 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 using this coordinate system. Our probe vehicle used for data acquisition is equipped with cameras, radar, and a LiDAR to mimic any autonomous vehicle, like the CUbE [32]. A detailed description of the sensor setup is shown in Figs. 3 and 4.

B. Localization Pipeline

1) Sensor Data Preprocessing : From the sensor data collected in the calibration run on the target roads, we first map LiDAR points collected by Velodyne VLP32 to images captured by the front Blackfly camera using the vehicle coordinate system at each time step. In other words, only LiDAR points that overlap with the field-of-view of the front-camera are used, as Fig. 4 depicts. We use the term *frame* to denote the images and the LiDAR points collected at each time step.

In Fig. 5, we show one example, where Fig. 5a is the original image captured by the front camera at a certain time step and Fig. 5b is the visualization of the LiDAR points mapping. From the images and the mapped LiDAR points, we next try to get objects which will be served as vertices in the graph topological landmark map introduced in Section II-B2. To this end, we use a semantic segmentation neural network based on DeepLabV3 [33] architecture. The model was trained on Cityscapes [34] dataset, and cropped instance segmentations are obtained from it. The cropped instance segmentations shown in Figs. 6b to 6d are examples gotten from the original image Fig. 5a and the semantic segmentation Fig. 6a. We resize all the cropped segmentations to size 256×256 with paddings. The static roadside objects, including traffic lights, traffic signs, poles, road edges, and building windows, are utilized in our localization framework, while other dynamic objects are discarded since they are expected to be anywhere in the second run on the target roads.

We call the cropped instance segmentations the *landmark segmentations* if they are the static roadside object snapshots. Let $s^{m,j}$ ($j \in \mathbb{N}^+$) denote each landmark segmentation in the road partition m. Let $p_{s^{m,j}}$ denote the points cloud reflected from landmark segmentation $s^{m,j}$. As well, $p_{s^{m,j}}$ can be obtained from the LiDAR points mapping and the pixel positions where the landmark segmentations are located in the original images.

We call the tuple $d^{m,j} = (s^{m,j}, p_{s^{m,j}})$ a *landmark object* in the road partition m, if $s^{m,j}$ is a landmark segmentation of the road partition. 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 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 vertices set $V^m = \{d^{m,j}\}_j$ contains the static roadside landmark objects in each partition m obtained from Section II-B2. 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 $\frac{\sum_{q \in p_{s^{m,j}}} q}{|p_{s^{m,j}}|}$ where $|p_{s^{m,j}}|$ is the the cardinality of $p_{s^{m,j}}$, and the distance between two objects is measured by the L_2 norm. The landmark map shown in

Fig. 7 is an example for illustration. For simplicity, in this report we only consider constructing the map by using the static objects from the frame collected at the reference point in each road partition. We will improve the map quality in the future by considering integrating objects from all frames collected in each partition. In this regard, we denote $G = \{\mathcal{G}^m\}_{m=1}^{M}$.

3) Real-time Localization:

(i) Get real-time measurements from sensors and construct the measurement graph:

At each time t, we utilize the data measured in real-time from vehicle sensors. Specifically, similar to the steps taken in the landmark map construction, we construct a measurement map $\mathcal{G}_t^x = (V_t^x, E_t^x)$ based on the frame collected at the time step t using the static roadside objects gotten from the image processed by the semantic segmentation neural network and the LiDAR points. 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 vertices to the vertex.

(ii) Perform a rough localization to determine which $\mathcal{G}_t^y \in G$ the vehicle is currently in:
For simplicity, we use GPS measured at each time step t to determine the rough location of the vehicle. We may use other techniques like EKF, image matching, etc. to do a rough localization when GPS is even not available. For a concrete illustration, see Fig. 9 and note that if the rough location from GPS at the time step t shows that the vehicle is in road partition $m+1$, we will have $\mathcal{G}_t^y = \mathcal{G}^{m+1}$.

(iii) 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 Graph Attention Networks:
We need to find out the common landmark objects in the landmark map graph $\mathcal{G}_t^y$ and the measurement graph $\mathcal{G}_t^x$. 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 common if $s_t^{x,i}$ and $s_t^{y,i}$ are snapshots of the same static roadside object. The reason for getting the common landmark objects will be clear in Item (iv) and Section II-D. Briefly speaking, our localization neural network (LNN) will compare the features extracted from the common landmark objects 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 contain 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 are collected from different locations with different road conditions. We design a landmark objects matching neural network (LOMNN) to perform matching between the vertices in $\mathcal{G}_t^x$ and $\mathcal{G}_t^y$. The network is detailed in Section II-C. When the matched pairs of landmark objects are found, the common landmark objects in two different graphs are dug out. We construct two sub-graphs containing the common landmark objects, 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 images, our matching method can not only exploit the images of landmark segmentations but also make full use of the LiDAR points on these segmentations.

(iv) Perform localization using LNN :

LNN can reason about the graph structured data $\mathcal{G}^{x}_{t,,\text{match}}$ and $\mathcal{G}^{y}_{t,\text{match}}$ by comparing the real-time measured data to the calibration landmark map. The output of LNN is the estimated vehicle's location with respect to (w.r.t.) the localization coordinate system in each road partition. LNN is introduced in details in Section II-D.

C. Landmark Objects Matching Based on Graph Attention Networks

Here, a matching method is proposed to reveal whether a given pair of landmark objects can be regarded as the same.

1) Matching Model Based on GAT: Let '1" and "0" denote the matched label and the mismatched label, representing whether the pair of landmark objects are matched or not.

As far as the model is concerned, it mainly consists of three kinds of neural networks, namely Resnet [13], GAT [35] and PointNet [14], which are respectively used for feature extraction of images, subgraph feature description, as well as feature comparison. The specific procedure is shown in Fig. 10, whose details are given as follows.

- **Step 1: Feature extraction for landmark objects with Resnet and PointNet.** First of all, two landmark objects $d_t^{x,i} = (s_t^{x,i}, p_{s_t^{x,i}})$ and $d_t^{y,j} = (s_t^{y,j}, p_{s_t^{y,j}})$ are chosen arbitrarily from $\mathcal{G}_t^x$ and $\mathcal{G}_t^y$. Then, the Resnets and PointNets are used to extract the features of $d_t^{x,i}$ and $d_t^{y,j}$. In particular, similar to the conventional image processing methods based on neural networks, the Resnets are adopted to extract the high dimensional features for the landmark segmentations $s_t^{x,i}$ and $s_t^{y,j}$, whose outputs are denoted by $R(s_t^{x,i})$ and $R(s_t^{y,j})$. Meanwhile, the PointNets are exploited to handle the corresponding LiDAR points cloud $p_{s_t^{x,i}}$ and $p_{s_t^{y,j}}$, whose outputs are denoted by $P(p_{s_t^{x,i}})$ and $P(p_{s_t^{y,j}})$. Thus, the extracted concatenated features of $d_t^{x,i}$ and $d_t^{y,j}$ are denoted by $f(d_t^{x,i})$ and $f(d_t^{y,j})$ respectively, where $f(d_t^{x,i}) = (R(s_t^{x,i})||P(p_{s_t^{x,i}}))$ and $f(d_t^{y,j}) = (R(s_t^{y,j})||P(p_{s_t^{y,j}}))$, as well as $||$ denotes the concatenation operation.
- **Step 2: Feature description with GAT.**
  First of all, with respect to the landmark objects $d_t^{x,i} = (s_t^{x,i}, p_{s_t^{x,i}})$ and $d_t^{y,j} = (s_t^{y,j}, p_{s_t^{y,j}})$, the corresponding neighborhood maps denoted by $\tilde{\mathcal{G}}_t^{x,i} = (\tilde{V}_t^{x,i}, \tilde{E}_t^{x,i})$ and $\tilde{\mathcal{G}}_t^{y,j} = (\tilde{V}_t^{y,j}, \tilde{E}_t^{y,j})$ are constructed, where $\tilde{V}_t^{x,i}$ and $\tilde{V}_t^{y,j}$ are the set of vertices, as well as $\tilde{E}_t^{x,i}$ and $\tilde{E}_t^{x,i}$ are the set of edges. Specifically, $\tilde{V}_t^{x,i}$ consists of the k nearest landmark objects $\{d_t^{x,l}\}_{l\in\mathcal{R}_k^i}$ to the $d_t^{x,i}$ (which is regarded as the i-th vertex in $\mathcal{G}_t^{x,i}$) and $\tilde{E}_t^{x,i}$ denotes the corresponding connection relationships matrix for these landmark objects. As well, $\mathcal{R}_k^i$ denotes the set of subscripts of k nearest vertices to the i-th vertex in $\mathcal{G}_t^{x,i}$. Similarly, $\tilde{V}_t^{y,j}$ and $\tilde{E}_t^{y,j}$ are the notations like $\tilde{V}_t^{x,i}$ and $\tilde{E}_t^{x,i}$. Then, the landmark objects in $\tilde{\mathcal{G}}_t^{x,i}$ and $\tilde{\mathcal{G}}_t^{y,j}$ are processed by Resnets and PointNets to extract the corresponding features. That is, the landmark segmentations in $\tilde{\mathcal{G}}_t^{x,i}$ and $\tilde{\mathcal{G}}_t^{y,j}$ are processed by means of Resnets and the corresponding LiDAR points cloud are handled by PointNets. Furthermore, the features of landmark objects and the corresponding connection relationship matrices in the $\tilde{\mathcal{G}}_t^{x,i}$ and $\tilde{\mathcal{G}}_t^{y,j}$ are handled by GAT to describe their features containing spatial information. Specifically, taking the graph $\tilde{\mathcal{G}}_t^{x,i}$ as an example [35], the attention mechanism of graph attentional layer w.r.t the pair of vertices (u, v) is described as

$$\xi_{uv} = \frac{\exp(\text{LeakyReLU}(\boldsymbol{a}^T[W \times f(d_t^{x,u})||W \times f(d_t^{x,v})]))}{\sum_{l\in\mathcal{R}_k^i}\exp(\text{LeakyReLU}(\boldsymbol{a}^T[W \times f(d_t^{x,u})||W \times f(d_t^{x,l})]))}, \ (u, v \in \mathcal{R}_k^i), \tag{1}$$

  where $(\cdot)^T$ denotes transposition, $||$ is the concatenation operation, ξ_{uv} is the attention coefficient of vertex v to u, W denotes the weight matrix for each vertex, as well as $\boldsymbol{a}$ is the weight vector. It is noted that LeakyReLU nonlinearity is used here. When the multi-head attention is considered, that is, M independent attention mechanisms are used to stabilize the learning process, the output features of each vertex (after applying a nonlinearity σ) are given by

$$f'(d_t^{x,u}) = \mathop{||}_{r=1}^{M} \sigma\Big(\sum_{v\in\mathcal{R}_k^i} \xi_{uv}^r \times W^r \times f(d_t^{x,v})\Big), \tag{2}$$

  especially, the output of the final (prediction) layer is described as

$$f'(d_t^{x,u}) = \sigma\Big(\frac{1}{M}\sum_{r=1}^{M}\sum_{v\in\mathcal{R}_k^i} \xi_{uv}^r \times W^r \times f(d_t^{x,v})\Big). \tag{3}$$

  At last, the whole output from GAT, namely $g(\tilde{\mathcal{G}}_t^{x,i})$, consists of all $f'(d_t^{x,u})$ $(u \in \mathcal{R}_k^i)$.
- **Step 3: Comparison based on neural networks.** In this step, outputs in the previous two steps are concatenated as the inputs for the PointNet which is regarded as the similarity measure for the feature comparison. Moreover, the Multi-Layer Perceptron (MLP) is introduced to output the predicted labels, where the loss function is the cross entropy loss, i.e.

$$L_{ce}(\boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}) = -\sum_i \alpha^{(i)} \log \hat{\alpha}^{(i)} + (1 - \alpha^{(i)}) \log(1 - \hat{\alpha}^{(i)}), \tag{4}$$

in which $\alpha^{(i)}$ and $\hat{\alpha}^{(i)}$ denote the true label and predicted label respectively.

- **Step 4: Matched sub-graph construction.** After finding out the common landmark objects in the landmark map graph $\mathcal{G}_t^y$ and the measurement graph $\mathcal{G}_t^x$, we construct two sub-graphs containing the common landmark objects, and denote them as $\mathcal{G}_{t,,\text{match}}^x$ and $\mathcal{G}_{t,\text{match}}^y$ respectively. Two vertices in $\mathcal{G}_{t,\text{match}}^x$ (respectively, $\mathcal{G}_{t,\text{match}}^y$) are connected to each other if and only if the two vertices are connected to each other in $\mathcal{G}_t^x$ (respectively, $\mathcal{G}_t^y$) .

2) Experiment Results and Discussion: In this subsection, the experiments are taken to evaluate the performance of the proposed matching method. The real data from Continental Automotive Pte Ltd is used to train and test the model.

Particularly, the landmark segmentations include traffic lights, traffic signs and lamp posts, where there are also a number of LiDAR points on them. As well, approximate $5,000$ pairs of segmentations are used, in which 80% pairs with labels are training data and 20% pairs are testing data. In the training process, there are 100 epochs for our model to reveal the corresponding matching performance. As for the criterions, Accuracy score, Recall score and F_1 score are considered to show the matching performance. In addition, different number for k nearest neighbors in neighborhood maps are also chosen in the proposed model so that we can see if number k has great impact on the matching performance.

From Fig. 11, it is illustrated that the proposed model has pretty good performance on landmark objects matching. With different k in this model, the corresponding Accuracy score, Recall score and F_1 score are almost over 98%. This also indicates that there is no too much improvement by using different number for the k nearest neighbors.

D. Localization Neural Network

Our LNN first uses $f^x(\cdot)$ and $f^y(\cdot)$ to extract features for landmark objects $d_{t,\text{match}}^{x,i} = (s_{t,\text{match}}^{x,i}, p_{s_{t,\text{match}}^{x,i}})$ and $d_{t,\text{match}}^{y,i} = (s_{t,\text{match}}^{y,i}, p_{s_{t,\text{match}}^{y,i}})$ in $\mathcal{G}_{t,\text{match}}^x$ and $\mathcal{G}_{t,\text{match}}^y$ respectively. $f^x(\cdot)$ and $f^y(\cdot)$ have the same network architecture but separate trainable parameters, which is depicted in Fig. 12 and Fig. 13. Similar to the operation in Section II-C1, Resnets $R^x(\cdot)$ and $R^y(\cdot)$ are used to extract features from the common landmark segmentations. Additionally, PointNets $P^x(\cdot)$ and $P^y(\cdot)$, which are applicable for reasoning about unordered data (e.g. LiDAR points) with arbitrary numbers (mainly because of the max-pooling operator in PointNet), extract features from $p_{s_{t,\text{match}}^{x,i}}$ and $p_{s_{t,\text{match}}^{y,i}}$. GAT $g^x(\cdot)$ will further 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 $E_{t,\text{match}}^x$, where $f^x(d_{t,\text{match}}^{x,i}) = (R^x(s_{t,\text{match}}^{x,i})||P^x(p_{s_{t,\text{match}}^{x,i}}))$ is the concatenated features of vertex i, as well as $||$ denotes the concatenation operation. The similar operation is performed by $g^y(\cdot)$ on $\{f^y(d_{t,\text{match}}^{y,i})\}_i$ and $E_{t,\text{match}}^y$. Here $f^x(\cdot)$ and $f^y(\cdot)$ also have the same network architecture but separate parameters. This the same case for $g^x(\cdot)$ and $g^y(\cdot)$. RNN (e.g. LSTM) can be applied here after GAT and will perform learning from measurements history $\{\mathcal{G}_{t'}^x\}_{t'=1}^{t-1}$ during the driving time. However, in the experimental evaluation results

presented in Section III, this RNN structure is left out for simplicity. We will do more tests with the RNN structure in the future. A new PointNet then performs a comparison for the extracted features from the pairs of common landmark objects. MLP is finally applied to perform the neural network regression using the output of the final features from this PointNet.

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

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

where g is the ground truth location and $\hat{g}$ is the model output. During the training of LNN, we intentionally perturb the ground truth matching of objects with small probability (less than 0.1), which makes the test result more promising.

III. Preliminary Test Results

We perform evaluations in two datasets generated by CARLA simulator and a real dataset provided by Continental Automotive Pte Ltd. Instead of the localization results in the real dataset, we get some odometry results using pairs of adjacent frames. This is because in the real dataset the labels are not fully provided. We will do an entire test in the future.

A. Results on Carla Dataset

We use CARLA simulator generate two different datasets with different driving traces in one town, as Fig. 14 depicts, namely CARLA dataset 1 and CARLA dataset 2.

We split the each dataset (around 5000 frames) for training with 80% frames and testing with 20% frames. We include traffic lights, poles, traffic signs, roadside objects (chairs, flowerpots, etc.) as landmark objects and set partition length $L = 2$ (metres). We first train our LNN using the training set in CARLA dataset 1 and test it using the testing set from the same CARLA dataset. The results are shown in Table II and Fig. 15.

TABLE II
Test results in Carla dataset 1 with $L = 2$.

X direction RMSE	Y direction RMSE	Inference time[1]
0.015m	0.066m	$0.3s$

We then use the LNN training in CARLA dataset 1 to perform testing in CARLA dataset 2. The results are shown in Table III. We observe that LNN performs slightly worse with some unseen data from a different driving trace. As well, the centimeter localization accuracy is achieved under the criterion of Root Mean Squared Error (RMSE). There are over 80% prediction results whose distance errors are less than 0.05 metre. Moreover, the mean of inference time is less than 0.5 second in the testing performance, which is acceptable in practice.

TABLE III
TEST RESULTS IN CARLA DATASET 2 WITH $L = 2$.

X direction RMSE	Y direction RMSE	Inference time
0.021m	0.078m	$0.3s$

B. Results on Real Dataset

We also test our model in the real dataset provided by Continental Automotive Pte Ltd and get some odometry results using pairs of adjacent frames. The test results are shown in Table IV and Fig. 16. From these, it is apparent that the centimeter localization accuracy is available for the real dataset under the criterion of Root Mean Squared Error (RMSE). There are over 90% prediction results whose distance errors are less than 0.05 metre. Moreover, the mean of inference time in the real dataset is acceptable in practice, which is similar to that in Carla dataset.

TABLE IV
ODOMETRY TEST RESULTS IN THE REAL DATASET.

X direction RMSE	Y direction RMSE	Inference time
0.001m	0.015m	$0.3s$

REFERENCES

[1] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments," in *Robotics: science and systems*, 2007, p. 1.

[2] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car—part i: Distributed system architecture and development process," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 12, pp. 7131–7140, 2014.

[3] I. Skog and P. Handel, "In-car positioning and navigation technologies—a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 4–21, 2009.

[4] 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 *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No. 04TH8720)*, vol. 2, 2004, pp. 772–789.

[5] 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 *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2012, pp. 1–6.

[6] H. Durrant-Whyte and T. C. Henderson, "Multisensor data fusion," in *Springer handbook of robotics*, 2016, pp. 867–896.

[7] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, 1999, pp. 1150–1157.

[8] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*, 2006, pp. 404–417.

[9] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*, 2006, pp. 430–443.

[10] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*, 2011, pp. 2564–2571.

[11] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[12] N. Engel, S. Hoermann, M. Horn, V. Belagiannis, and K. Dietmayer, "Deeplocalization: Landmark-based self-localization with deep neural networks," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 926–933.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[15] A. Elfes *et al.*, "Occupancy grids: A stochastic spatial representation for active robot perception," in *Proceedings of the Sixth Conference on Uncertainty in AI*, vol. 2929. Morgan Kaufmann, 1990, p. 6.

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

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

[18] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *J. Comput., vol*, vol. 7, no. 3, 1978.

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

[20] H. Lategahn and C. Stiller, "Vision-only localization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pp. 1246–1257, 2014.

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

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

[23] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard, "Efficient estimation of accurate maximum likelihood maps in 3d," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 3472–3478.

[24] M. A. Paskin, "Thin junction tree filters for simultaneous localization and mapping," in *in Int. Joint Conf. on Artificial Intelligence*. Citeseer, 2003.

[25] H. F. Durrant-Whyte, "Uncertain geometry in robotics," *IEEE Journal on Robotics and Automation*, vol. 4, no. 1, pp. 23–31, 1988.

[26] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[28] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[29] A. Valada, N. Radwan, and W. Burgard, "Deep auxiliary learning for visual localization and odometry," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6939–6946.

[30] L. Ding and C. Feng, "Deepmapping: Unsupervised map estimation from multiple point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8650–8659.

[31] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[32] A. Hartmannsgruber, J. Seitz, M. Schreier, M. Strauss, N. Balbierer, and A. Hohm, "Cube: A research platform for shared mobility and autonomous driving in urban environments," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2315–2322.

[33] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[34] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[35] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

Claims

1. Method for determining a position of a vehicle, wherein the method comprises the steps:

providing a reference map, the reference map comprising segmentations of a reference image with landmarks;

capturing real time LIDAR data and a measurement image of a vehicle environment;

determining segmentations of the measurement image and neighborhood graphs, where a segmentation is represented by a vertex and where a neighborhood graph comprises the vertex and edges containing information to identify neighboring vertices of the vertex, to obtain a measurement map

comparing segmentations of the reference map with the segmentations represented by the vertices of the measurement image and the neighborhood graphs and determining segmentations contained in the reference map and in the measurement image; and

estimating the vehicle's position with reference to the reference map during its movement along a road based on a result of the comparison.

2. Method according to claim 1, wherein the step comparing segmentations of the reference map with the segmentations represented by the vertices of the measurement image and the neighborhood graphs comprises the step:

performing a rough localization to determine a road partition, which the vehicle is currently in, and selecting a reference image from the reference map that is related to the road partition.

3. Method according to claim 1 or 2, wherein the step comparing segmentations of the reference map with the segmentations represented by the vertices of the measurement image and the neighborhood graphs comprises selecting an object from a set of objects contained in an image; wherein a segmentation represents an object.

4. Method according to any of the previous claims, wherein the step comparing segmentations of the reference image with the segmentations

represented by the vertices of the measurement image and the neighborhood graphs comprises:

extracting features from a segmentation of a selected object by a first Resnet, and extracting features from the LiDAR data of the selected object by a first PointNet and providing the extracted features to a common PointNet.

5. Method according to claim 4, wherein the step of comparing segmentations of the reference map with the segmentations represented by the vertices of the measurement image and the neighborhood graphs comprises additionally:

extracting features from neighbor segmentations of the segmentation of the selected object and providing the extracted features to a GAT,

extracting features from LiDAR points cloud of a neighboring object of the selected object and providing the extracted features to the GAT, and

describing the extracted features containing spatial information, and providing the described reference image features to the common PointNet.

6. Method according to claim 5, wherein the steps of claims 3 to 5 are performed for the reference image and the measurement image; wherein the method further comprises the step:

concatenating the extracted features from the first ResNet, the first PointNet and the GAT and determining a similarity between the features of the object of the reference image and the features of the object of the measurement image.

7. Method according to claim 6, wherein the step determining a similarity between the features of the object of the landmark map network part and the features of the object of the measurement map part comprises predicting labels by a Multi-Layer Perceptron (MLP) and calculating a loss function by calculating a cross entropy.

8. Method according to any of the previous claims, wherein the step providing a reference map, the reference map comprising segmentations with landmarks, comprises the sub-steps:

capturing LIDAR data points and a reference image along a road for a road partition;

mapping the LIDAR data points to the reference image;

determining objects on the reference image and determining landmark segmentations from the reference image using a semantic segmentation neural network; and

constructing a graph topological landmark map containing vertices corresponding each to a segmentation and edges, wherein an edge identifies neighboring vertices of a vertex.

9. Computer program element for determining a position of a vehicle, which, when running on a processing unit, causes the processing unit to carry out any of the methods as defined in claims 1 to 8.

10. Computer program element for determining matching landmark objects, the computer program element comprising:

a landmark map network part and measurement map part; each of the landmark map network part and measurement map part comprising:

an object selection module configured to select an object from a set of objects contained in an image;

a Resnet configured to extract features from a first segmentation of the selected object and providing the extracted features to a common PointNet,

a PointNet configured to extract features from a first LiDAR point cloud of the selected object and providing the extracted features to the common PointNet,

a second Resnet configured to extract features from neighbor segmentations of the first segmentation and providing the extracted features to a GAT,

a second PointNet configured to extract features from LiDAR points cloud of the neighboring object of the selected object and providing the extracted features to a GAT,

the GAT, configured to describe the extracted features containing spatial information, and to provide the described landmark map features to the common PointNet;

the common PointNet configured to concatenate the extracted features from the first ResNet, the first PointNet and the GAT and to determine a similarity between the features of the object of the landmark map network part and the features of the object of the measurement map part.

11. Computer readable medium on which a computer program element according to claim 9 or 10 is stored.

Abstract

The invention relates to a method for determining a position of a vehicle is provided. The method comprises the following steps: In a first step, a reference map is provided. The reference map comprises segmentations of a reference image with landmarks. In a second step, real time LIDAR data and a measurement image of a vehicle environment are captured. In a third step, segmentations of the measurement image and neighborhood graphs are determined to obtain a measurement map, wherein a segmentation is represented by a vertex and where a neighborhood graph comprises the vertex and edges containing information to identify neighboring vertices of the vertex. In a fourth step, segmentations of the reference image are compared with the segmentations represented by the vertices of the measurement image and the neighborhood graphs and segmentations contained in the reference image and in measurement image are determined. In a fifth step, estimating the vehicle's position with reference to the reference map during its movement along a road based on a result of the comparison is carried out.

(Figure 10)