FLAME: Feature-Likelihood Based Mapping and Localization for Autonomous Vehicles

Su Pang, Daniel Kent, Daniel Morris and Hayder Radha

***Abstract*— Accurate vehicle localization is arguably the most critical and fundamental task for autonomous vehicle navigation. While dense 3D point-cloud-based maps enable precise localization, they impose significant storage and transmission burdens when used in city-scale environments. In this paper, we propose a highly compressed representation for LiDAR maps, along with an efficient and robust real-time alignment algorithm for on-vehicle LiDAR scans. The proposed mapping framework, which we refer to as Feature Likelihood Acquisition Map Emulation (FLAME), requires less than 0.1% of the storage space of the original 3D point cloud map. In essence, FLAME emulates an original map through feature likelihood functions. In particular, FLAME models planar, pole and curb features. These three feature classes are long-term stable, distinct and common among vehicular roadways. Multiclass feature points are extracted from LiDAR scans through feature detection. A new multiclass-based point-to-distribution alignment method is proposed to find the association and alignment between the multiclass feature points and the FLAME map. The experimental results show that the proposed framework can achieve the same level of accuracy (less than 10cm) as the 3D point cloud based localization.**

I. Introduction

A fundamental task for autonomous vehicles is to accurately determine its position at all times. Multiple key sub-systems rely either fully or partially on the performance of the localization algorithm. It has been estimated that decimeter level localization accuracy is required for autonomous vehicles to drive safely and smoothly [1]. GNSS-based (Global Navigation Satellite System) techniques struggle to achieve this level of accuracy except for open sky areas [2], [3]. Map-based localization frameworks, especially those that utilize Light Detection and Ranging (LiDAR) based localization methods [4], [5], are popular because they can achieve centimeter level accuracy regardless of light conditions. However, a key drawback of any localization method that relies on 3D point cloud maps is the enormous size of the map itself. Consequently, there is a need for efficient representations of such maps while maintaining high-accuracy localization capabilities. The representation format should contain sufficient information for vehicles to localize and be lightweight enough to be stored and downloaded into vehicles in real-time when needed. Furthermore, it is important to note that environments do change rather frequently, and it is therefore important to have the ability to update the map to reflect these changes.

Su Pang, Daniel Kent, Daniel Morris and Hayder Radha are with the Department of Electrical and Computer Engineering, College of Engineering, Michigan State University, 220 Trowbridge Road, East Lansing, Michigan, 48824, United States. Email: (pangsu@msu.edu, kentdan3@egr.msu.edu, dmorris@msu.edu, radha@egr.msu.edu)

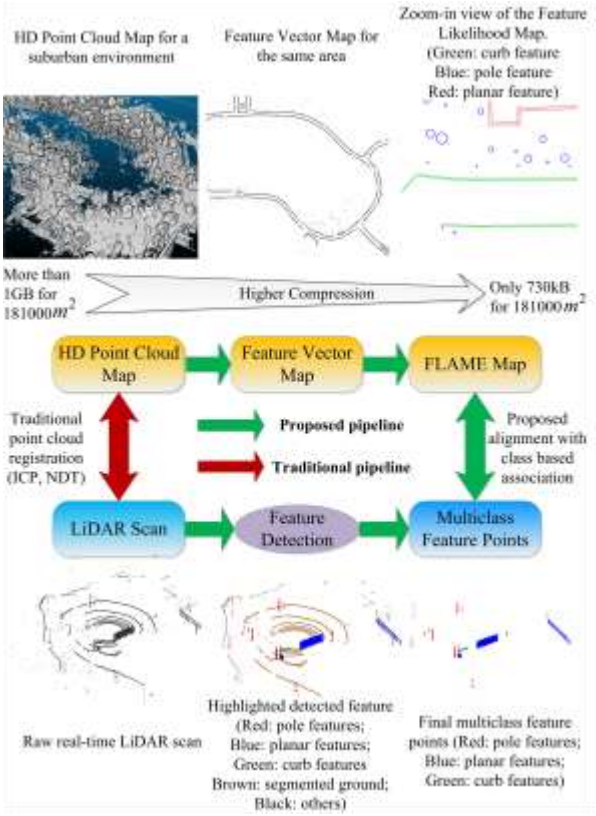

Figure 1: Overview of our proposed FLAME map-based localization architecture.

In this paper, we propose a lightweight, map-based localization framework, Feature Likelihood Acquisition Map Emulation (FLAME). Fig. 1 shows an overview of our system architecture. From the map point of view, instead of the point cloud, compressed and lightweight features are used to represent the environment. As made clear later, FLAME emulates an original map through feature likelihood functions. The size of the final FLAME map is under 0.1% of the original point cloud. To align the real-time LiDAR scan to the feature map for localization, corresponding features are extracted in real-time from the scan through feature detection. Experimental results show that our proposed method can achieve comparable centimeter level localization accuracy to the traditional dense 3D point cloud based methods. The main contributions are:

- A novel FLAME map representing a 3D urban point cloud in a highly compressed form.
- A real-time, feature-detection method for on-vehicle

LiDAR scans.

- A multiclass feature association and alignment algorithm that achieves accurate alignment (error less than 10cm) between the detected LiDAR scan features and the FLAME map.

The paper is organized as follows: Section II introduces related works. Section III illustrates the FLAME system architecture. Section IV and section V explain the processing of map and feature detection respectively. Section VI describes the alignment between detected features and feature map. Section VII shows the experimental results and finally in section VIII, we conclude the paper.

II. Related Work

An overview of the related work on localization in 3D point clouds was discussed in [6] and [7]. In this section, we only review feature map-based methods.

Many efforts have explored different methods to compress city-scale maps while ensuring accurate localization. During the DARPA urban challenge, all teams were provided with a digital street map of the environment in the form of a Road Network Definition File (RNDF) and a high-resolution aerial image of the test site which can be used to enhance the RNDF for localization and planning [8]–[11]. The RNDF file is the prototype of the HD vector map that professional mapping companies offer for autonomous driving nowadays. Most of the teams used a fused localization system based on GPS, odometry, inertial measurement unit (IMU) and LiDARs [8]–[11]. These methods can determine the global pose in lane level accuracy. Philipp Ruchti *et al.*[12] proposed a method to localize a mobile robot based on OpenStreetMap [13] and 3D LiDAR. They developed a road classification scheme for 3D LiDAR data and a novel sensor model, which relates the classification results to a road network. However, their localization accuracy is only meter-level. Domonique Gruyer *et al.*[14] provide an ego-lane level of accuracy localization system using an accurate digital lane marking map and cameras. Using only the lane markings in the digital map limits the application scenarios for the system since lanes can be blocked by other vehicles and not well painted in some roadways. Pole features are long-term stable and very common in the environment. A decimeter-level accuracy pole-based localization for autonomous vehicles using stereo camera system is proposed in [15]. E. Javanmardi *et al.*[16] proposed a multilayer 2D vector map based localization framework using 3D LiDAR. The only class of features contained in their 2D vector map is building footprints; and hence, alignment would fail for urban regions where mapped building facades are not within range of the scanning LiDAR.

Recently, neural networks have been leveraged to enable learning-based localization and mapping [17]–[20]. These include a learnable data-driven 3D segment descriptor to encode the point cloud, [17] and [18], which localize using segment feature matching and geometric verification, and the extracted features reconstruct the environment. A. Zaganidis *et al*. [19] used *PointNet*, a deep neural network for extracting semantic information from point cloud and incorporate it in point cloud registration. These methods learn features from the training data instead of handcrafting them; they have future potential but suffer from computation, robustness and accuracy issues.

III. System Architecture

Fig. 1 shows the overview of the system architecture and dataflows. The green arrows in Fig. 1 describe the proposed pipeline. From the map end, three classes of features, planar, pole and curb, are extracted from the original point cloud map and make up the feature vector map. Then the proposed FLAME map is generated based on the uncertainty of the feature vector map. From the LiDAR sensor end, corresponding classes of features are extracted from the real-time LiDAR scan through the feature detection algorithm. Then the proposed multiclass based association alignment algorithm is used to align the multiclass feature points and the FLAME map for localization.

IV. FLAME Map

A. Classes of Features

We propose a three-class feature set to represent environments around vehicular roadways. The feature set is chosen so that features are common and broadly representative, as well as being discriminative and readily localized. As mentioned before, we consider three feature classes: *Planar features*, which include walls, building facades, fences, signs with large boards, and other planar surfaces. *Pole features* that include the pole part of streetlights, traffic signs, tree trunks and similar vertical shapes. Finally, *curb features* discretely sample curbs. Each of these feature types is easily detected by LiDAR, even for low resolution LiDAR, and can be modeled by simple geometric entities, as described in the next subsection.

B. Generating 2D Feature Vector Map

Our feature map is automatically created from an HD 3D point cloud map. The 3D point cloud map data used in this study was collected by a mobile mapping system from a third party equipped with a 32 lines LiDAR, a precise GNSS, IMU and several cameras with an algorithm similar to GraphSLAM [21]. The resolution of the raw point cloud map data is usually more than 400 points/m^2.

The first step is to remove ground pixels and organize above-ground pixels into cells. The points above the ground plane are extracted from the point cloud using Cloth Simulation Filter [22]. This is followed by rasterizing the off-ground points into small cells on the X-Y plane with resolution of r by r. The cell c_i can be represented using a vector: $c_i = \{\boldsymbol{P}_{c_i}, N_{c_i}\}$, where $\boldsymbol{P}_{c_i}$ represents the point set located within this cell, and N_{c_i} is the number of points in $\boldsymbol{P}_{c_i}$. From our experience, a resolution should be at most 0.2m by 0.2m to obtain centimeter level accuracy in localization.

Next, our three feature classes are extracted from the rasterized point cloud. All the landmark features are tall vertical features above the ground plane, which leads to the fact that the cells containing landmark features would comprise more points when compared to other cells. Hence, we filter out the cells c_i such that $N_{c_i} < N_t$, where N_t is a threshold that is a function the original point cloud density and the resolution of rasterizations. A 2D feature point map

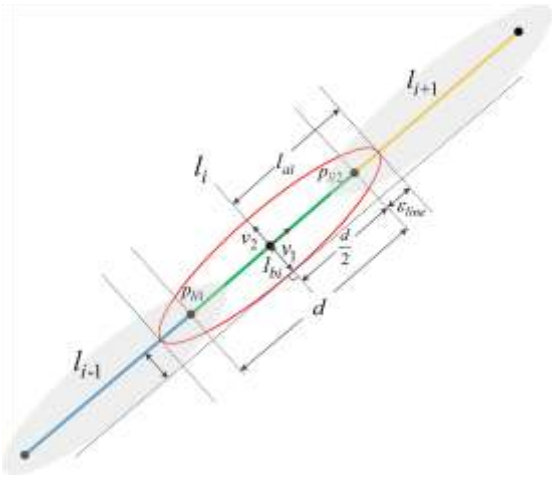

Figure 2: The calculation of mean value μ_i and covariance cov_i for the likelihood field of line segment l_i. The green line represents line segment l_i, the blue line and yellow line stand for the line segment l_{i-1} and line segment l_{i+1} respectively. The red ellipse is the 99% confidence ellipse generated from $N(\boldsymbol{\mu}_i, cov_i)$.

can be formed using the *x* and *y* coordinates of the remaining cells' center point. Planar features and curb features can be described as a set of connected line segments in 2D. From this feature point map, connected line segments representing planar features can be extracted using Random Sample Consensus (RANSAC) or Hough transform. Using the same method for the segmented ground points, one can extract the curb features. As for pole features, since the radius of most pole features in the urban and suburban environments are less than 20cm, which can be approximately modeled as a point in 2D. Applying 2D point clustering and calculating the center of the cluster could extract the point-based pole features from the point feature map.

The extracted three feature classes constitute the 2D feature vector map.

C. Generating FLAME Map

For traditional point-cloud registration-based localization, the real-time LiDAR scan is matched to a point cloud map through a point-to-point matching process. However, to perform feature matching effectively, the feature vector map built in the previous section needs to be converted into feature likelihood map (FLAME map) represented with normal distributions as described in this section. Compared to a feature vector map, a FLAME map can better emulate the original point cloud map. Under FLAME, normal distribution likelihood field functions can be treated as the generative process for the actual points from the landmark features in the original point cloud map.

The parameters for each normal distribution likelihood field are calculated as follows. For planar features modeled as connected line segments in the feature vector map, we divide the connected line segments into smaller line segments of length d. A normal distribution is used to model each line segment l_i as a likelihood field with a mean value $\boldsymbol{\mu}_i$ and a covariance matrix cov_i. The mean value $\boldsymbol{\mu}_i$ is the center point of this line segment. The calculation of cov_i is as follows. First, as shown in Fig. 2, assuming there is a 99% confidence ellipse for $N(\boldsymbol{\mu}_i, cov_i)$ along this line segment, i.e. $\boldsymbol{\mu}_i$ is the center of the confidence ellipse, the line segment direction is the major axis direction. The underlying meaning of this confidence ellipse is that 99% of the points generated by the normal distribution $N(\boldsymbol{\mu}_i, cov_i)$ would locate within this ellipse. The length of the major axis l_{ai} and minor axis l_{bi} of the confidence ellipse would be as follows:

$$l_{ai} = 2\sqrt{s \cdot \lambda_1} = 2\left(\frac{d}{2} + \varepsilon_{line}\right) \tag{1}$$

$$l_{bi} = 2\sqrt{s \cdot \lambda_2} = 2\left(\frac{\frac{d}{2} + \varepsilon_{line}}{10}\right) \tag{2}$$

where s is the 99% Chi-Square likelihood with 2 degrees of freedom, λ_1 and λ_2 are the larger and smaller eigen value of the covariance matrix cov_i respectively. ε_{line} is the uncertainty factor. Therefore, λ_1 and λ_2 can be calculated from equations (1) and (2). The positive direction of the major and minor axis of the confidence ellipse would be the same as the direction of the eigenvector corresponding to λ_1 and λ_2 respectively. Therefore, we can build the unit eigenvectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ according to the line segment direction. As shown in Fig. 2, $\boldsymbol{v}_1$ would point to the same direction as this line segment, $\boldsymbol{v}_2$ would be vertical to $\boldsymbol{v}_1$. So, for a line segment l_i, the corresponding likelihood filed modeled by normal distribution $N(\boldsymbol{\mu}_i, cov_i)$ can be calculated as follows:

$$\boldsymbol{\mu}_i = \frac{(\boldsymbol{p}_{li1} + \boldsymbol{p}_{li2})}{2} \tag{3}$$

$$cov_{l_i} = \boldsymbol{V} \cdot \boldsymbol{D} \cdot \boldsymbol{V}^{-1} \tag{4}$$

$$\boldsymbol{V} = [\boldsymbol{v}_1 \quad \boldsymbol{v}_2], \boldsymbol{D} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{5}$$

where p_{li1} and p_{li2} are the end points of line segment l_i. $\boldsymbol{V}$ is the orthonormal eigenvector matrix.

For curb features, we use the same processing method because curb features are also represented as connected line segments in the feature vector map. The only difference is that we use a larger uncertainty factor ε_{curb}. This is the case since when compared to walls, fences and building facades, curbs are much lower to the ground and usually not strictly vertical to the ground, and this would result in a larger uncertainty.

Poles are represented as points in the feature vector map, and their feature likelihoods are modeled by normal distribution $N(\boldsymbol{\mu}_{pole_j}, cov_{pole_j})$, $\boldsymbol{\mu}_{pole_j}$ is the coordinate of $pole_j$, cov_{pole_j} can be calculated as follows:

$$cov_{pole_j} = \frac{r^2}{s} \cdot \boldsymbol{I} \tag{6}$$

where r represents the radius of the confidence circle, s is the 99% Chi-Square likelihood with 2 degrees of freedom, $\boldsymbol{I}$ is 2x2 identity matrix. If the radius of the pole is provided, r in this equation can be changed into $r_{pole_j} + \varepsilon_{pole}$, where r_{pole_j} is the radius of the pole and ε_{pole} is the uncertainty factor.

V. Feature Detection

Up to this point, a dense 3D point cloud of the environment has been transformed into a FLAME map. Our goal is ego localization using real-time LiDAR scans. However, rather than directly matching 3D LiDAR points to the feature map, we propose first transforming these scans into feature maps themselves and then aligning them for ego localization. As we describe in this section, this has the advantages of improved association with fewer ambiguities as well as reduced computation, while simultaneously maintaining high accuracy.

Finding features in a LiDAR scan differs from feature detection in a dense, pre-scanned point cloud due primarily to the decrease in resolution with range from sensor. We select feature detectors that can operate at a broad range of resolutions, including low sampling resolution, and describe them in this section. The pole and planar features detectors are taken from [23] and summarized below, while we develop our own curb feature detector.

A. Scan Preprocessing

LiDAR points are initially filtered to obtain surface normals and are categorized as ground, steep and other. Surface normals are estimated from neighboring pixels in a depth image as described in [23]. These normals provide a steepness measure for pixels; those close to vertical (within 12.5 degrees) are labeled steep pixels, and are used to gather evidence for our pole and planar features. Any appropriate ground detection method could be used [24]–[27] to label ground pixels.

B. Pole Feature Detection

We search for pole features by first dividing the LiDAR scan into a regular array of cells of size *1*m. Within each cell we gather evidence for a pole and its location. Poles are modeled as vertical cylinders with LiDAR pixels distributed on the surface of the pole and with normal vectors that point away from the pole center and within 90 degrees of the sensor. Pole parameters, $\theta_{pole} = (x_j, y_j, r_j)$, are its center location and radius. Assuming a cell contains a pole, a LiDAR pixel with location $\boldsymbol{x}_i$ in a cell and normal $\boldsymbol{n}_i$ can predict a pole center as illustrated in Fig 3. This prediction, $P(\boldsymbol{x}_i, \boldsymbol{n}_i | T_{pole}, \theta_{pole})$, can be modeled as Gaussian centered at the pole center, and sampled over grid centers.

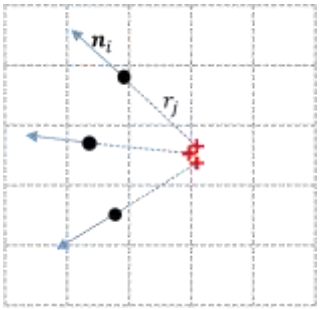

Figure 3: An illustration of evidence gathering for pole detection, in this case radius r_j. A cell is divided into a uniform grid at 10cm spacing. Each steep LiDAR point (black dot) predicts a pole center offset r_j along its normal. Predictions are interpolated into the grid, and a Hough Transform finds the maximum likelihood pole center, with a search over multiple radii.

If the cell is not a pole cell, then model $P(\boldsymbol{x}_i, \boldsymbol{n}_i | T_{nonpole})$ as a uniform density over grid centers equal to $\frac{1}{n_g}$, where n_g is the number of grid elements in each cell.

Assuming independent pixel samples, we can solve for the maximum likelihood ratio of pole to non-pole, $MLR_{pole}(\theta_{pole})$, using a probabilistic Hough transform [28]. This maximizes the following expression over pole locations and radii:

$$MLR_{pole}(\theta_{pole}) = \underset{\theta_{pole}}{\operatorname{argmax}}\, n_g^N \prod_{i=1}^{N_s} P(x_i, n_i | T_{pole}, \theta_{pole}) \quad (7)$$

Here N_s is the number of steep pixels, and N the number of non-ground pixels. The result is a pole estimate for each cell along with a likelihood ratio for a pole existing there. A threshold on this likelihood ratio is determined discriminatively. Further details of the method are in [23].

C. Planar Feature Detection

Planar features, like pole features, are detected using an array of cells distributed over the LiDAR scan. In each cell, a planar feature is modeled as a set of steep pixels with normals having the same orientation in the horizontal plane, θ_{wall}. Non-planar features are modeled as having uniform normal orientation. Once again a probabilistic Hough transform solves for the maximum likelihood ratio of wall to non-wall analogous to (7), along with a wall orientation θ_{wall}, with further details in [23].

D. Curb Detection

Curbs are another class of common and important features that are readily observable by moving vehicles. Based on the geometry and topology of the curb features around the vehicle, they can provide reliable constraints in the lateral direction. There are many curb detection algorithms available[29]–[31], but these methods are computationally expensive, and work best with a combination of high resolution 3D LiDAR and other sensors. This paper proposes a lightweight curb detection method, which can be used for any resolution LiDAR.

We have experimented the proposed method with a low-resolution LiDAR that returns a point cloud formed by 16 concentric measurements, which we refer to as rings. Further, the LiDAR is mounted on the top of the vehicle and parallel to the ground. If the height is fixed, for empty flat ground, the distance between each consecutive ring can be determined. Points where consecutive rings are closer than this are candidates for being curbs.

We compute the distance between rings for flat ground as illustrated in Fig. 4. The i^{th} ring radius is:

$$r_i^{regular} = \frac{h}{\tan\theta_i} \quad (8)$$

where h is the LiDAR height relative to the flat ground and θ_i is the laser beam angle. The distance between ring i and i+1 is:

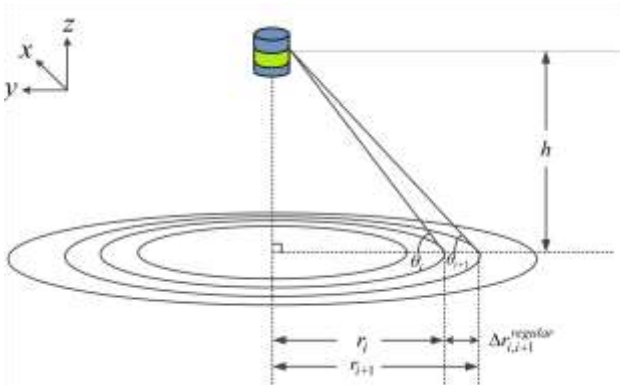

Figure 4: Scenario that 3D LiDAR beams intercept a flat plane.

$$\Delta r_{i,i+1}^{regular} = r_{i+1} - r_i \,, 0 < i < \mathrm{n} - 1 \tag{9}$$

Additional factors influence the distance between consecutive rings including non-flat ground and the LiDAR is not being mounted exactly parallel to the ground. To accommodate these we define an interval:

$$d_{i,i+1} = \left[a\Delta r_{i,i+1}^{regular}, b\Delta r_{i,i+1}^{regular}\right], \quad 0.5 < a < b < 1 \tag{10}$$

where a and b are parameters that adjust the range bounds. Points with $\Delta r_{i,i+1}$ falling within $d_{i,i+1}$ are categorized as curb candidates.

To avoid misclassifying sloped regions as curbs we develop the following strategy. We observe that gradients provide useful cue: curbs typically have a large height variance in lateral direction (y direction) and low height variance in longitudinal direction (x direction). To quantify this, the LiDAR scan is placed in an n by m cell grid, where n is the number of rings and each row i stores a ring. Each cell grid $c_{i,j}$ stores the mean value $(x_{i,j}, y_{i,j}, z_{i,j})$ of the position of points that fall in $c_{i,j}$. Therefore, the gradient is calculated as follows:

$$g_{i,j} = \frac{z_{i,j+1} - z_{i,j-1}}{\mathrm{dist}_{xy}(c_{i,j+1}, c_{i,j-1})} \tag{11}$$

where $z_{i,j}$ is the z value of the element in i^{th} row and j^{th} column in the cell grid. $\mathrm{dist}_{xy}(a, b)$ represents the Euclidean distance between a and b in x-y plane. For each grid $c_{i,j}$, we compute $g_{i,j}$ and compare it with a threshold t_g to classify it as a curb candidate or not. Finally, based on the ground segmentation depicted in the previous subsection, only points classified both as ground and curb are classified as curb points.

VI. Association and Alignment

The two key components of ego localization are feature association and feature alignment. This section proposes a localization method that leverages the three separate feature types: pole, planar and curb. Similar to NDT, which optimizes a score function related to the position density for each feature rather than requiring explicit association [32], we build the score for each detected feature of the current scan by summing densities from nearby map features of the same class and optimize this. Our full method is summarized in Algorithm 1.

The state of the system is defined as $\boldsymbol{X}_t = (x_t, y_t, \theta_t)^T$, a three-parameter pose in 2D. $\boldsymbol{z}_t^{pole}$, $\boldsymbol{z}_t^{planar}$ and $\boldsymbol{z}_t^{curb}$ stand for the detected pole, planar and curb points from the LiDAR scan at time t. *FLAME_map* contains $N4$ likelihood fields, $\boldsymbol{m}_i$, which each encompassing the class name, mean value $\boldsymbol{\mu}_i$ and the covariance matrix cov_i. v_t^m and θ_t^m are the measured vehicle linear speed and heading angle at time stamp t from odometry and IMU used for motion prediction.

Line 2 shows the motion prediction using the previous state, linear vehicle speed v_t^m and heading angle θ_t^m. T is the sampling period. The motion model is only used for providing the initial guess to initialize the iterative alignment. The whole iterative alignment process is depicted in lines 3~22.

Taking pole class feature as example (*c=='pole'* in Algorithm 1), in line 8, for a transformed pole feature point $T(\widehat{\boldsymbol{X}}_t, \boldsymbol{p}_i^{pole})$, the correspondence is determined by finding the top 3 closest pole feature likelihood fields in the map. $T(\boldsymbol{X}, \boldsymbol{p})$ is defined as a function that transforms a point $\boldsymbol{p}$ by pose $\boldsymbol{X}$. $\widehat{\boldsymbol{X}}_t$ contains motion prediction error, and hence, associating $T(\widehat{\boldsymbol{X}}_t, \boldsymbol{p}_i^{pole})$ to the closest map pole feature likelihood field may not be correct. However, among the top three closest map pole feature likelihood fields, the probability of containing the correct correspondence would be much higher. Therefore, the strategy is to find all the potential correspondences and

Algorithm 1: Multiclass Based Association and Alignment

1	*Variables:* $\boldsymbol{z}_t^{pole} = \{\boldsymbol{p}_1^{pole}, \boldsymbol{p}_2^{pole}, \dots, \boldsymbol{p}_{N1}^{pole}\}$, $\boldsymbol{z}_t^{planar} = \{\boldsymbol{p}_1^{planar}, \boldsymbol{p}_2^{planar}, \dots, \boldsymbol{p}_{N2}^{planar}\}$, $\boldsymbol{z}_t^{curb} = \{\boldsymbol{p}_1^{curb}, \boldsymbol{p}_2^{curb}, \dots, \boldsymbol{p}_{N3}^{curb}\}$, $\boldsymbol{p}_i^{class} = (x_i^{class}, y_i^{class})$, $class = \{$*'pole', 'planar', 'curb'*$\}$, $FLAME\_map = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3, \dots, \boldsymbol{m}_{N4}\}$, $\boldsymbol{m}_i = \{class, \boldsymbol{\mu}_i, cov_i\}$, $\boldsymbol{X}_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1})^T$
2	$\widehat{\boldsymbol{X}}_t = \begin{bmatrix}\hat{x}_t \\ \hat{y}_t \\ \hat{\theta}_t\end{bmatrix} = \begin{bmatrix}x_{t-1} + v_t^m \cdot T \cdot cos\theta_t^m \\ y_{t-1} + v_t^m \cdot T \cdot sin\theta_t^m \\ \theta_t^m\end{bmatrix}$ *%Motion Prediction*
3	***while*** *not converged* ***do***
4	***for*** *each c* ***in*** *class:*
5	$\boldsymbol{score}^c \leftarrow \boldsymbol{0}$
6	$\boldsymbol{g}^c \leftarrow \boldsymbol{0}$
7	$\boldsymbol{H}^c \leftarrow \boldsymbol{0}$
8	***for*** *all transformed detected c feature points* $T(\widehat{\boldsymbol{X}}_t, p_i^c)$
9	***do*** *kd search with radius* $\leq r$ *to find the top 3 closest features in likelihood_map with class == c*
10	*Update the negative score s (see equation (12))*
11	$score^c \leftarrow score^c + s_i^c(\widehat{\boldsymbol{X}}_t)$
12	*Update the gradient vector* $\boldsymbol{g}^c$
13	*Update the Hessian Matrix* $\boldsymbol{H}^c$
14	***end for***
15	***end for***
16	$score^{all} = score^{pole} + score^{planar} + score^{curb}$
17	$\boldsymbol{g}^{all} = \boldsymbol{g}^{pole} + \boldsymbol{g}^{planar} + \boldsymbol{g}^{curb}$
18	$\boldsymbol{H}^{all} = \boldsymbol{H}^{pole} + \boldsymbol{H}^{planar} + \boldsymbol{H}^{curb}$
19	***solve*** $\boldsymbol{H}^{all} \cdot \Delta\widehat{\boldsymbol{X}}_t = -\boldsymbol{g}^{all}$
20	$\widehat{\boldsymbol{X}}_t \leftarrow \widehat{\boldsymbol{X}}_t + \Delta\widehat{\boldsymbol{X}}_t$
21	***end while***
22	***return*** $\boldsymbol{X}_t \leftarrow \widehat{\boldsymbol{X}}_t$

combine them into the score. If fewer than three pole feature likelihood fields are found, only those found are used. If there is no pole feature likelihood field found in the neighborhood, then the algorithm drops this detected pole point in this iteration as it may be a false positive. This strategy also makes the system robust to dynamic moving objects that are not modeled in the map. For example, passing vehicles that may be detected as planar features should not be associated to any feature likelihood field in the FLAME map.

Then, the score for a feature of class c is calculated and updated in lines 10~11 by summing the scores of up to 3 nearby map features of the same class:

$$s_i^c(\widehat{\boldsymbol{X}}_t) = \sum_{j=1}^{3} exp\left(-\frac{1}{2}\left(T(\widehat{\boldsymbol{X}}_t, \boldsymbol{p}_i^c) - \boldsymbol{\mu}_j\right)^T \cdot cov_j^{-1} \cdot \left(T(\widehat{\boldsymbol{X}}_t, \boldsymbol{p}_i^c) - \boldsymbol{\mu}_j\right)\right). \quad (12)$$

The total score over all features is found as described in lines 12~13, as well as the gradient vector and Hessian Matrix of the pose vector $\widehat{\boldsymbol{X}}_t$.

In the proposed framework, the search for correspondence is constrained to the same class of features; for example, the detected curb feature points can only be associated with the curb feature likelihood fields in the map. This multiclass-based association significantly reduces false alignments. The negative score, gradient vector and Hessian matrix are calculated based on their classes separately, but are all parameterized with the vehicle pose $\widehat{\boldsymbol{X}}_t$. Newton's method is used to maximize the total score function (lines 17~20).

To reduce the complexity of the system, the proposed architecture only focuses on three-parameter pose estimation in two dimensions. This is sufficient for most driving environments where the vehicle remains on the ground surface.

VII. Experiments and Analysis

A. Experimental Platform and Test Sites

The experimental platform used in this study is a modified, drive-by-wire Lincoln MKZ. The vehicle is equipped with a 16 lines LiDAR (VLP-16), and a NovAtel PwrPak7 GNSS Inertial Navigation System to collect the near-ground truth data with an error of 2~3cm for analysis. A Core i7 CPU runs Ubuntu and Robot Operating System (ROS).

Two test sites in typical suburban environment included in this study. Test site #1 is a 62000 m^2 purpose-built proving ground, University of Michigan's MCity Test Facility. Test site #2, as shown in Fig. 5, is West Circle Drive, a traffic hub in Michigan State University. Note that both the two test sites are roughly flat with no steep slopes.

B. Evaluation and Comparison

To evaluate the performance of the proposed localization framework, we drove the vehicle for several loops with different speeds in the test sites and evaluated the position estimation by comparing the proposed method with traditional 3D point cloud registration based localization (NDT driven using the Point Cloud Library) and other related methods. Noted that the dynamic objects are not removed in the process of building 3D point cloud map and the real time LiDAR scans in our experiments. TABLE I shows the comparison of position estimation mean absolute error (MAE) and orientation MAE error between the proposed method and other methods. We performed two types of experiments. One is an overall localization experiment; the second is a single-step alignment experiment. For overall localization, as shown in Algorithm 1 in Section 6, it is a recursive algorithm, the current state $\boldsymbol{X}_t$ is updated from the previous state $\boldsymbol{X}_{t-1}$. In the single-step alignment experiment, the current state $\boldsymbol{X}_t$ is updated from the previous ground truth state. For some of the methods we used for comparison, they have relatively large errors in each step of the alignment; and consequently, the accumulative error for each step would make the recursive algorithm fail and impossible to evaluate quantitatively. Using the previous ground truth to initialize the update would keep the algorithm running and make it feasible to evaluate and compare the error in each single step of the alignment for the whole route. The error for position estimation is defined as the Euclidean distance between the position estimation and the near-ground truth. The orientation error is defined as the absolute difference between the estimated yaw angle and the near-ground truth yaw angle. TABLE I shows that the proposed method can achieve centimeter-level localization accuracy and same level of performance as the traditional 3D point cloud registration based method. Performance was good even for road segments when only two of the three feature classes were observed.

TABLE II shows the comparison of the size for the two types of maps. The proposed FLAME map uses far less memory and computation than a traditional point cloud map. According to [33], 3D point cloud map with resolution of 121 points/m^2 could achieve the best localization performance using VLP-16. This result illustrates that for localization applications, many of the rich details in 3D point cloud maps contribute little to improving localization accuracy, and can be efficiently encoded as class labels for feature likelihoods.

The use of multi-class features that are associated and aligned within class is advantageous when features are in close proximity. This is particularly the case when maps are projected into a 2D plane. Ignoring class can lead to incorrect associations. Fig. 6 compares the proposed multi-class based association alignment with non-class based alignment. This illustrates how non-class based alignment results in curb

Figure 5: Test site #2, West Circle Drive at the campus of Michigan State University. The length of the route is about 1.45 km. The red line highlights the driven route.

TABLE I. COMPARISON OF POSITION ESTIMATION (POS) MAE ERROR AND ORIENTATION (YAW) MAE ERROR BETWEEN PROPOSED METHOD AND OTHER METHODS.

Experiment Type	Map type	LiDAR data	Exp #1 (5~10mph)				Exp #2 (15~20mph)				Exp #3 (25~30mph)			
			Test Site#1		Test Site#2		Test Site#1		Test Site#2		Test Site#1		Test Site#2	
			Pos	Yaw	Pos	Yaw	Pos	Yaw	Pos	Yaw	Pos	Yaw	Pos	Yaw
Single-step Alignment	3D point cloud	LiDAR scans	4.05	0.29	7.13	0.35	5.10	0.42	7.58	0.38	5.02	0.41	6.31	0.45
	Non-class FLM	2D LiDAR scans	17.83	1.58	22.83	2.04	19.47	1.75	21.39	1.99	20.17	1.69	20.83	2.16
	Non-class FLM	Non-class feature points	12.54	0.92	10.37	0.88	14.52	1.36	11.35	0.95	14.10	1.13	12.94	1.03
	FLAME map	**Multiclass feature points**	8.23	0.577	7.45	0.46	8.10	0.62	7.93	0.59	8.31	0.49	7.96	0.61
Overall Localization	3D point cloud	LiDAR scans	4.06	0.34	7.42	0.40	5.42	0.46	7.61	0.40	5.14	0.46	6.83	0.48
	FLAME map	**Multiclass feature points**	8.76	0.92	8.15	0.46	8.31	0.86	9.47	0.61	9.13	0.79	8.75	0.64

Different type of LiDAR data has been tested to align with different type of maps. FLM in this table stands for Feature Likelihood Map. Non-class FLM and non-class feature points are the feature data without class semantic information, such as planar, pole or curb. The unit for position estimation and yaw angle are centimeter and degree respectively. The method framed by bold lines is the proposed method. For single-step alignment experiment, the current state $\boldsymbol{X}_t$ is updated from the previous near-ground truth state.

features being miss-associated with planar-features on walls. On the other hand, multi-class features avoid this issue leading to improved alignment.

For map-based localization algorithms, environmental changes, which are not modeled in the prior map, can affect localization performance. Fig. 7 illustrates one scenario with some detected dynamic and static objects that are not included in the map. The measured LiDAR points from the side of the bus are detected as planar features. The huge bus also blocks the field of view behind it, so the features behind it cannot be detected. Despite this, the proposed multiclass based association alignment framework works well as no features are associated with the bus features.

TABLE II. COMPARISON OF THE SIZE BETWEEN PROPOSED FLAME MAP AND 3D POINT CLOUD MAP

Map Type \ Test Site		Test Site #1 (62000 m^2)	Test Site #2 (181000 m^2)
FLAME Map		210.1 kB	730 kB
3D Point Cloud Map	36 points/m^2	35.4 MB	87.6 MB
	121 points/m^2	152.3 MB	230.2 MB
	400 points/m^2	403.3 MB	580.0MB

VIII. CONCLUSION

We introduced a new mapping and localization framework that is based on distinct feature classes and corresponding feature likelihood functions. The proposed FLAME map multiclass features enable robust localization, which could be achieved despite the fact that the required space needed for representing FLAME maps are orders of magnitude smaller than traditional 3D point cloud maps. Furthermore, a new feature-to-feature alignment algorithm is proposed to align the detected features from the LiDAR scans to the FLAME map. Using intra-class association improves alignment robustness when features are dense. Experimental results show that the proposed method can achieve centimeter level localization accuracy and is robust to static and dynamic changes in real environments that are not modeled in the map.

In future work, we plan to extend the FLAME approach to different sensor modalities and additional feature detection algorithms. Furthermore, we will evaluate our methods on some public available datasets collected with different sensors, such as KITTI [34] and nuScene [35] dataset. Moreover, it could of interest to incorporate graph-based optimization as the backend and build a complete online SLAM framework based on FLAME.

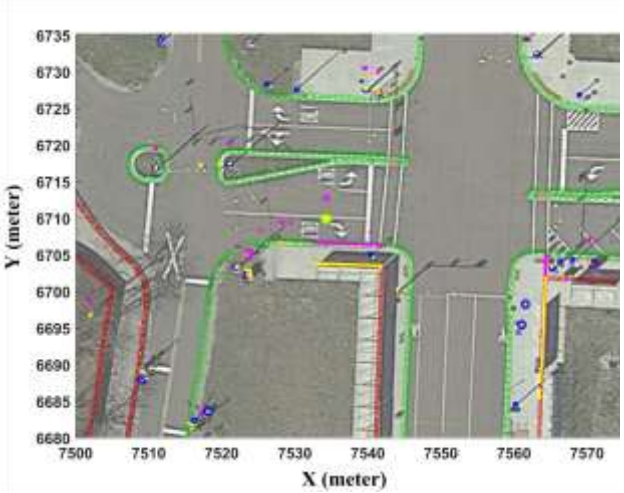

Figure 6: Comparison between the proposed multiclass based association alignment and non-class association alignment. The green, red and blue ellipses are the feature likelihood fields for curb, planar and pole features respectively. The pink points are the wrong alignment result using the non-class association method. The yellow points are the alignment result using the proposed method. The pink star, yellow star and green triangle are the position estimation based on non-class based alignment, the proposed method and near-ground truth position respectively.

REFERENCES

[1] J. Levinson, M. Montemerlo, and S. Thrun, "Map-Based Precision Vehicle Localization in Urban Environments.," in *Robotics: Science and Systems*, 2007, vol. 4, p. 1.

[2] R. Yozevitch, B. Ben-Moshe, and A. Dvir, "GNSS accuracy improvement using rapid shadow transitions," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 3, pp. 1113–1122, 2014.

[3] Y. Gu, L.-T. Hsu, and S. Kamijo, "GNSS/onboard inertial sensor integration with the aid of 3-D building map for lane-level vehicle self-localization in urban canyon," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 4274–4287, 2016.
[4] R. W. Wolcott and R. M. Eustice, "Robust LIDAR localization using multiresolution Gaussian mixture maps for autonomous driving," *Int. J. Robot. Res.*, vol. 36, no. 3, pp. 292–319, 2017.
[5] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 4372–4378.
[6] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, 2016.
[7] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 829–846, 2018.
[8] M. Montemerlo *et al.*, "Junior: The stanford entry in the urban challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, 2008.
[9] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," in *The DARPA Urban Challenge*, Springer, 2009, pp. 1–59.
[10] J. Leonard *et al.*, "A perception-driven autonomous urban vehicle," *J. Field Robot.*, vol. 25, no. 10, pp. 727–774, 2008.
[11] S. Kammel *et al.*, "Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 615–639, 2008.
[12] P. Ruchti, B. Steder, M. Ruhnke, and W. Burgard, "Localization on openstreetmap data using a 3d laser scanner," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 2015, pp. 5260–5265.
[13] "https://www.openstreetmap.org." .
[14] D. Gruyer, R. Belaroussi, and M. Revilloud, "Map-aided localization with lateral perception," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, 2014, pp. 674–680.
[15] R. Spangenberg, D. Goehring, and R. Rojas, "Pole-based localization for autonomous vehicles in urban scenarios," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 2016, pp. 2161–2166.
[16] E. Javanmardi, M. Javanmardi, Y. Gu, and S. Kamijo, "Autonomous vehicle self-localization based on multilayer 2D vector map and multi-channel LiDAR," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 437–442.
[17] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based place recognition in 3d point clouds," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5266–5272.
[18] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: 3D Segment Mapping using Data-Driven Descriptors," in *Robotics: Science and Systems (RSS)*, 2018.
[19] A. Zaganidis, L. Sun, T. Duckett, and G. Cielniak, "Integrating deep semantic segmentation into 3-D point cloud registration," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 2942–2949, 2018.
[20] P. Wang, R. Yang, B. Cao, W. Xu, and Y. Lin, "Dels-3d: Deep localization and segmentation with a 3d semantic map," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5860–5869.
[21] S. Thrun and M. Montemerlo, "The graph SLAM algorithm with applications to large-scale mapping of urban structures," *Int. J. Robot. Res.*, vol. 25, no. 5–6, pp. 403–429, 2006.
[22] W. Zhang *et al.*, "An easy-to-use airborne LiDAR data filtering method based on cloth simulation," *Remote Sens.*, vol. 8, no. 6, p. 501, 2016.
[23] D. D. Morris, "Obstacles and foliage discrimination using lidar," in *Unmanned Systems Technology XVIII*, 2016, vol. 9837, p. 98370E.
[24] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, 2010, pp. 560–565.
[25] B. Douillard *et al.*, "On the segmentation of 3D LIDAR point clouds," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 2798–2805.
[26] T. Chen, B. Dai, R. Wang, and D. Liu, "Gaussian-process-based real-time ground segmentation for autonomous land vehicles," *J. Intell. Robot. Syst.*, vol. 76, no. 3–4, pp. 563–582, 2014.
[27] M. Zhang, D. D. Morris, and R. Fu, "Ground segmentation based on loopy belief propagation for sparse 3d point clouds," in *2015 International Conference on 3D Vision (3DV)*, 2015, pp. 615–622.
[28] R. S. Stephens, "Probabilistic approach to the Hough transform," *Image Vis. Comput.*, vol. 9, no. 1, pp. 66–71, 1991.
[29] T. Chen, B. Dai, D. Liu, J. Song, and Z. Liu, "Velodyne-based curb detection up to 50 meters away," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*, 2015, pp. 241–248.
[30] C. Fernández, D. F. Llorca, C. Stiller, and M. A. Sotelo, "Curvature-based curb detection method in urban environments using stereo and laser," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 579–584.
[31] W. Zhang, "LIDAR-based road and road-edge detection," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, 2010, pp. 845–848.
[32] P. Biber and W. Strasser, "The normal distributions transform: a new approach to laser scan matching," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, 2003, vol. 3, pp. 2743–2748 vol.3.
[33] P. Su, K. Daniel, C. Xi, A.-Q. Hothaifa, M. Daniel, and R. Hayder, "3D Scan Registration Based Localization for Autonomous Vehicles – A Comparison of NDT and ICP under Realistic Conditions," presented at the IEEE Connected and Automated Vehicles Symposium 2018, Chicago, 2018.
[34] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.
[35] H. Caesar *et al.*, "nuScenes: A multimodal dataset for autonomous driving," *ArXiv Prepr. ArXiv190311027*, 2019.