

POU-SLAM: Scan-to-Model Matching Based on 3D Voxels

Jianwen Jiang ¹, Jikai Wang ¹, Peng Wang ² and Zonghai Chen ¹

¹Department of Automation, University of Science and Technology of China, Hefei 230027, PR China

²Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield S1 3JD, United Kingdom

Abstract—Purpose: Localization and mapping with LiDAR data is a fundamental building block for autonomous vehicles. Though LiDAR point clouds can often encode the scene depth more accurate and steadier compared with visual information, laser-based Simultaneous Localization And Mapping (SLAM) remains challenging as the data is usually sparse, density variable and less discriminative. The purpose of this paper is to propose an accurate and reliable laser-based SLAM solution.

Design/methodology/approach: The method starts with constructing voxel grids based on the 3D input point cloud. These voxels are then classified into three types to indicate different physical objects according to the spatial distribution of the points contained in each voxel. During the mapping process, a global environment model with Partition of Unity (POU) implicit surface is maintained and the voxels are merged into the model from stage to stage, which is implemented by Levenberg-Marquardt algorithm.

Findings: We find a laser-based SLAM method. The method uses POU implicit surface representation to build the model and is evaluated on the KITTI odometry benchmark without loop closure. Experimental results indicate that the method achieves accuracy comparable to the state-of-the-art methods.

Originality/value: We propose a novel, low-drift SLAM method that falls into a scan-to-model matching paradigm. The method, which operates on point clouds obtained from Velodyne HDL64, is of value to researchers developing SLAM systems for autonomous vehicles.

Keywords—Simultaneous Localization And Mapping; voxel grids; scan-to-model; Partition of Unity

I. INTRODUCTION

Reliable localization of autonomous vehicles is an attractive research topic since it is a basic requirement for navigation and other tasks [1,2,3,4,5]. Fully autonomous vehicles should reliably locate themselves, ideally by using only their own sensors on board without relying on external information sources such as GPS. In scenarios where such external information is absolutely unavailable, Simultaneous Localization And Mapping (SLAM) can always be an alternative option.

SLAM has been intensively studied during the past few decades and various solutions based on different sensors have been proposed for both indoor and outdoor environments. In outdoor scenes, the advantage of LiDAR with respect to cameras is that the noise associated with each measurement is independent of the distance itself and usually more robust to the illumination variation. Therefore, laser-based SLAM is becoming one of the mainstream solutions in outdoor scenes. Most laser-based SLAM methods tend to extract stable features such as planes and edges from points, and then do SLAM

in a ‘feature space’. They are therefore categorized as feature-based SLAM, whose performance is mainly determined by two factors. The first factor is the way features are designed. Ji Zhang et.al [6,7] propose a planar and edge feature designing method based on curvature. Michael Bosse et.al [8] incorporate shape information which is calculated from geometric statistics of the point cloud into the Iterative Closest Point (ICP) correspondence step. The second factor is the scan matching method. Scan-to-scan and scan-to-model matching are the two main scan matching frameworks in SLAM. Sen Wang et.al [9] propose a scan-to-scan odometry trajectory estimation method by using convolutional neural networks to process LiDAR point clouds. KinectFusion [10] implements scan-to-model matching by using an implicit surface as model. The state-of-the-art laser-based SLAM method, Lidar Odometry And Mapping (LOAM) [6] extracts distinct features corresponding to physical surfaces and corners. To enable on-line implementation, LOAM switches between scan-to-scan at 10 Hz and scan-to-model operation at 1 Hz update frequency. Although LOAM has achieved good performance, there are still challenges in outdoor applications using 3D laser sensors, i.e., (1) inherent point matching error since the sparsity of the point clouds, which means that there are always existing errors when we directly use the sparse point cloud to fit the surface of environment. Furthermore, it is nearly impossible to obtain rigid point correspondences between scans; (2) it is hard to fix the matching parameter since the density variation of the point clouds. During the feature-based matching process, it is required to extract enough context information of one 3D point to make it discriminative, while the scope of the dominating context is greatly affected by the point density. In this paper, to tackle the above challenges, we introduce POU implicit surface representation to regress the environment surface with sparsity points, which can effectively encode the detailed geometrical information. We discretize the point cloud into 3D voxels. These voxels are classified into planar and edge feature voxels according to the geometrical character of the points contained in the voxel. Finally, we build a feature voxel map for further scan-to-model matching, during which the POU implicit surface representation is proposed to adapt to the voxel map. The key contributions of our work are as follows:

- (1) We propose a novel feature voxel map which stores voxels with salient planar or edge features.
- (2) We propose a scan-to-model matching framework using

POU implicit surface representation.

The rest of this paper is organized as follows. In Sect. II, we discuss related work and conclude how our work is different from the state of the art. The feature extraction algorithm is presented with details in Sect. III, and the scan-to-model matching algorithm is given in Sect. IV. Experimental results are shown in Sect. V. Finally, Sect. VI concludes the paper.

II. RELATED WORK

SLAM with cameras and LiDAR has attracted wide attention both from robotic and computer vision communities. We acknowledge the large body of work in this field, but concentrate here only on approaches based on LiDAR. Since feature extraction, laser-based mapping and scan-matching framework are the three most important steps for laser-based SLAM, we therefore look into related work on these three aspects and provide general ideas on how they work as following.

Feature extraction: LOAM focuses on edge and planar features in the LiDAR sweep and keep them in a map for edge-line and plane-planar surface matching. Tixiao Shan et.al [11] also use edges and planar features to solve different components of the 6-DOF transformation across consecutive scans. While these works extract features based on curvature, there is still difference like the method from [11] is ground-optimized, as it leverages the presence of a ground plane in its segmentation and optimization steps. Using point cloud obtained from a high resolution and high density Zoller-Fröhlich Z+F laser, J Lalonde and Yungeun Choe et.al [12,13] exploit point cloud geometrical statistics to classify the natural terrain into scatter-ness, linearity, and surface-ness. Using point cloud from a SICK LMS291 laser range sensor, Michael Bosse et.al [8] implement scan-matching based on point cloud geometrical statistics features. They use ICP to align voxel centroids, which can reduce the number of points for scan matching as well as the size of the map. Different from [8], we consider the geometric features of the points more subtly and implement scan matching with original points, which would not lead to information loss.

Laser-based mapping: SLAM systems often build and maintain environment map like feature map [6,7,11,14,15], dense map [16,17], subsampled clouds [18,19] and Normal Distributions Transform (NDT) based map [20,21]. In laser-based SLAM systems, feature map is usually a structural collection of surfaces and corners that are extracted from original point clouds. In this paper, we build a feature map, which can be distinguished from literature methods in the way that we maintain more types of features in the form of voxels.

Scan matching framework: Recently, some laser-based SLAM systems use scan-to-model matching framework and have achieved state-of-the-art performance on KITTI odometry benchmark. Jens Behley et.al [16] propose a dense approach to laser-based SLAM that operates on point clouds obtained from Velodyne HDL-64. They construct a surfel-based map and use a point-to-plane ICP [22,23] to estimate the pose transformation of the vehicle. Our method essentially belongs to scan-to-model matching framework. The distinction is that we build a feature voxel map and use POU implicit surface

representation to adapt to the scan-to-model matching process based on feature voxel map.

Partition of unity approach: The POU approach has been widely used in surface reconstruction. Ireneusz Tobor et.al [24] show how to reconstruct multi-scale implicit surfaces with attributes, given discrete point sets with attributes. Tung-Ying Lee et.al [25] propose a new 3D non-rigid registration algorithm to register two multi-level partition of unity implicit surfaces with a variational formulation.

III. FEATURE EXTRACTION

Similar to [26], the scan-to-model matching is triggered when there are salient geometrical features in local regions. So, it is important to find these regions first. In contrast to 3D object classification [27,28] and place recognition [29,30], which use complex point cloud descriptor to segment and classify objects, we use a simpler shape parameterization because our task focuses on incremental transformation for which we have a stronger prior on the relative poses.

A. Computing Shape Parameters

A local region that is occupied by the input point cloud is repeatedly divided until the number of points fall into each voxel is equal to the threshold N_p . The threshold N_p is related to the horizontal and vertical resolution of Lidar point cloud. In the process of feature extracting, the densities of features in corresponding voxels vary. We consider the density while searching for the correspondences of the features. The lower the density of the features, the lower the probability of finding enough correspondences within a certain radius is. Our experiment results are achieved when N_p is set to 25. Each voxel contains points that fall into it and voxel centroid. The points that fall into voxel are defined as $\{X_i\} = \{(x_i, y_i, z_i)^T\}$. The first and second-order moments μ, S describe the parameters for spatial distribution of the points $\{X_i\}$:

$$\mu = \frac{1}{N} \sum_{i=1}^n X_i \quad (1)$$

$$S = \frac{1}{N} \sum_{i=1}^n (X_i - \mu)(X_i - \mu)^T \quad (2)$$

Inspired by [8], the matrix S is decomposed into principal components ordered by increasing eigenvalues. $\vec{e}_0, \vec{e}_1, \vec{e}_2$ are eigenvectors corresponding to the eigenvalues $\lambda_0, \lambda_1, \lambda_2$ respectively, where $\lambda_0 \geq \lambda_1 \geq \lambda_2$. In the case that the structure of points in a voxel is a linear structure, the principal direction will be the tangent at the curve, with $\lambda_0 \gg \lambda_1 \approx \lambda_2$. In the case that the structure of points in a voxel is a solid surface, the principal direction is aligned with the surface normal with $\lambda_0 \approx \lambda_1 \gg \lambda_2$. The two saliency features, named linear feature and surface feature, are linear combinations of the eigenvalues. Figure 2 illustrates the two features used. The quantity:

$$c = \frac{\lambda_0 - \lambda_1}{\lambda_0 + \lambda_1 + \lambda_2} \quad (3)$$

is the linear feature of the points in voxel which ranges from 0 to 1. Similarly the quantity:

$$p = \frac{\lambda_1 - \lambda_2}{\lambda_0 + \lambda_1 + \lambda_2} \quad (4)$$

is the surface feature of the points in voxel which ranges from 0 to 0.5.

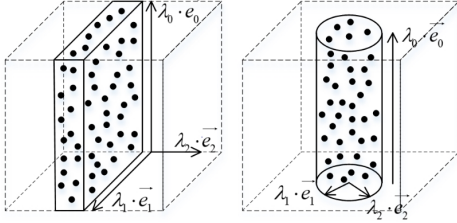


Fig. 1. Illustration of the surface feature and linear feature.

B. Voxel Sampling Strategy

Essentially, the voxel based scan matching process is to construct the associations between voxels. It is believed that voxels with high quantity of linearity or planarity tend to be more stable than others. So for each input point cloud, we classify its 3d grids according to the proposed quantities into three types: edge voxel, planar voxel and others. Furthermore, it should be noticed that the ground points usually contain more planar features and non-ground points usually contain both planar and edge features. With this bears in mind, we first divide the whole point cloud into ground and non-ground segments using the method proposed in [31]. Figure 3(a) is an example of the ground segmentation result. Then, for non-ground points, we extract planar and edge features, while only planar features are extracted for ground points.

In each scan which is now represented as the voxel centroids, we sort the voxels based on their linearity values c and surface-ness values p and get two sorted lists. Two thresholds c_{th} and p_{th} are then employed to distinguish different types of features. We call the voxels with c larger than c_{th} edge features, and the voxels with p larger than p_{th} planar features. Then n_{F_e} edge features with the maximum c are selected from each scan. Non-ground and ground planar features with the maximum p are also selected, and the numbers of the two types of features are n_{F_p} and $n_{F_{gp}}$, respectively. Edge and planar features extracted from all the scans are denoted as F_e , F_p and F_{gp} thereafter. Visualization examples of these features are given in Figure 3(b)-(d).

IV. SCAN-TO-MODEL MATCHING WITH POU IMPLICIT SURFACE REPRESENTATION

Once the current laser scan is transformed into a set of stable voxels, we implement scan-to-model matching process. In our method, the map consists of the last n located feature sets F_e , F_p and F_{gp} . Let

$$M^k = \{ \{M_e^k\}, \{M_p^k\}, \{M_{gp}^k\} \} \quad (5)$$

be the map that contains planar features and edge features at time k . To better fit the observed surface with these points

and adapt to the feature voxel map, we take the POU implicit surface representation to construct the model.

A. Finding Feature Point Correspondence

In the current scan, each point is labeled according to the type of its corresponding voxel. During the matching process, we construct the correspondences between voxels in the map and the current feature points with the guidance of labels and Euclidean distance. Since such correspondences generating mechanism narrows down the potential candidates for correspondences, it can help to improve the matching accuracy and efficiency. The detailed correspondence construction process is given below.

For each current edge point in feature set F_e , we search for n_e voxels of

$$M_e^k = \{F_e^{k-n}, \dots, F_e^{k-2}, F_e^{k-1}\} \quad (6)$$

inside a sphere space whose radius is r_e . For each corresponding voxel, we find an edge line as the correspondence of the current edge point. After finding all the correspondence edge lines, we can get the projection points S_e of the current edge point on the corresponded edge lines. Then we use linear quantity c to determine whether S_e can be represented as an edge line. If S_e can be represented as an edge line, it is then regarded as the correspondence of the current edge point. After the edge feature correspondences are obtained, we compute the distance from an edge feature point to its correspondence. The procedure of finding an edge line and the distance computation can be found in [6].

For each current planar point in feature sets F_p and F_{gp} , we respectively find n_p and n_{gp} voxels of

$$M_p^k = \{F_p^{k-n}, \dots, F_p^{k-2}, F_p^{k-1}\} \quad (7)$$

and

$$M_{gp}^k = \{F_{gp}^{k-n}, \dots, F_{gp}^{k-2}, F_{gp}^{k-1}\} \quad (8)$$

inside a sphere space whose radius is r_p as the corresponding voxels. Then, we use the corresponding voxels to construct the POU implicit surface representation of the model, and compute the distance from a planar feature point to its implicit surface representation of the model. The detail of building the POU implicit surface representation and compute the distance from a planar feature point to its implicit surface representation of the model will be discussed below.

B. POU Implicit Surface Representation For Planar Features

1) *POU Approach*: The basic idea of the POU approach is to break the data domain into several pieces, approximate the data in each subdomain separately, and then blend the local solutions together using smooth, local weights that sum up to one everywhere in the domain. More specifically, we assign a specific planar patch as the correspondence of a planar feature point in each correspondence voxel. The procedure of finding the specific points to form planar patches P_c can be found in [7]. We can calculate distances d between the planar feature point and the corresponding planar patches P_c . Then, we project planar feature point x on the surface of planar

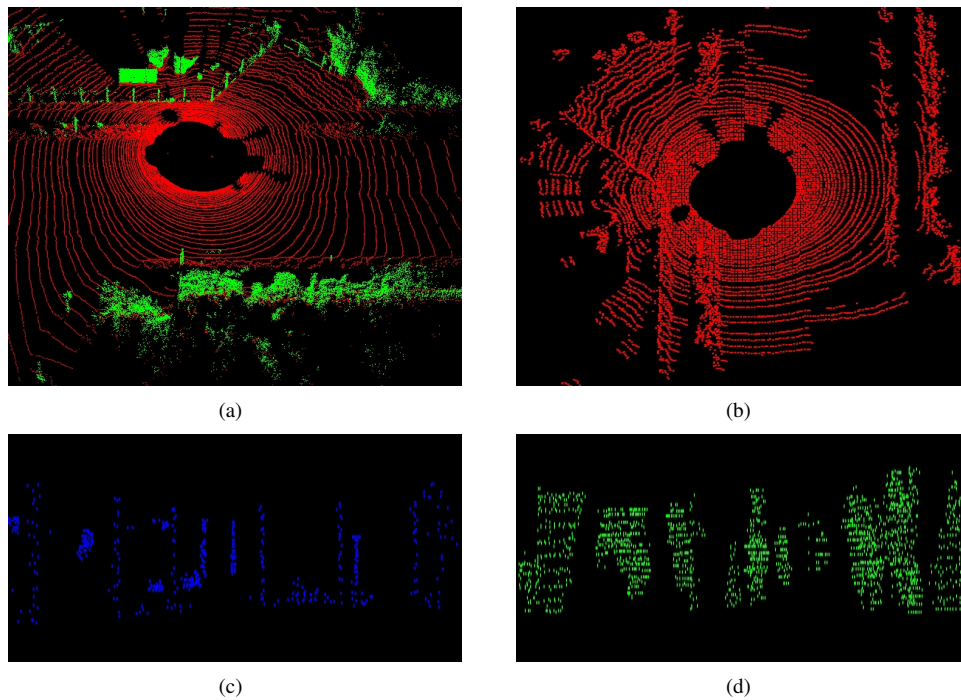


Fig. 2. (a) is the result of ground segmentation. In (a), the red points are labeled as ground points and the green points are labeled as non-ground points. (b)(c)(d) are the ground planar features, non-ground edge features and non-ground planar features.

patches defined by p_s : $p_{s_i} = x - d_i n_i$, where n_i is the normal of the closest point to x in each planar patch and is a good approximation of the surface normal at the projected point p_{s_i} . Each planar patch can be regarded as a subdomain. Then, we use the weight function to blend the subdomains together.

2) *POU Implicit Surface Representation*: The implicit surface is defined as the set of zeros of a function $I(x)$, which also behaves as a distance function from x to the implicit surface. In this work, we use the POU implicit surface representation.

Similar to the POU implicit surface framework in [32], we define the POU implicit function $I^{M^k}(x)$ as an approximate distance of planar feature point $x \in \mathbb{R}^3$ at time k to the POU implicit surface defined by the map M^k :

$$I^{M^k}(x) = \frac{\sum_{s_i \in P_c} W_{s_i}(x)(x - p_{s_i}) \cdot \vec{n}_{s_i}}{\sum_{s_j \in P_c} W_{s_j}(x)} \quad (9)$$

where p_{s_i} is the projected points on the corresponding planar patch s_i , \vec{n}_{s_i} is the normal at point p_{s_i} . For approximation, we use the quadratic B-spline $b(t)$ to generate weight functions for planar patches.

$$W_{s_i}(x) = b\left(\frac{3|x - c_{s_i}|}{2R_{s_i}}\right) \quad (10)$$

where c_{s_i} is the geometric center of planar patch s_i and $R_{s_i} = \max_{x \in s_i} \|x - c_{s_i}\|$ is a spherical support radius of the planar patch s_i . The function is proportional to the distance between x and c_{s_i} and inversely proportional to R_{s_i} . It means that the closer c_{s_i} to the x , the higher weight will be allocated. And the more scattered the points which form the planar patch are, the higher the weight. In addition to computing the distance from the feature point to the model surface, the surface normal at the projected point of the feature point x should also be computed.

In this work, we use the normal of the closest point $p_{s_{closest}}$ to x as the approximation of the surface normal at the projected point. Compared with the method using only one plane to represent a surface (could be cylinder or sphere etc.), we adopt multiple local planes with various planarities to represent a surface which can achieve higher accuracy and more accurate distances from feature points to their correspondences.



Fig. 3. Illustration of POU implicit surface. The Figure shows the POU implicit surface representation. Two cells, A and B , are associated with their support radius R_A and R_B , respectively. The value of a point x in the slashed region can be evaluated by $I^{M^k}(x) = \frac{W_A(x)Q_A + W_B(x)Q_B}{W_A(x) + W_B(x)}$; $W_A(x) = b(\frac{3d_1}{2R_A})$; $W_B(x) = b(\frac{3d_2}{2R_B})$, where d_1 and d_2 are the distances from the point x to the centers of the cell A and B , respectively.

3) *Motion Estimation*: With distances from feature points to their correspondences in hand, we assign a bisquare weight [7] for each feature point. The rules are twofold. In general, when the distance between the feature point and its correspondence is below a certain threshold, the weight is assigned inversely proportional to the distance. However, when the distance is greater than the threshold, the feature point is regarded as an outlier. We proceed to recover the LiDAR motion by minimizing the overall distances.

Algorithm 1 POU-SLAM Scan To Model Matching framework**Input:** Current Scan, Model**Output:** Calculate Lidar Pose Transform T Between Current Scan And Model

1. Feature Extraction

<Extract non ground features F_p^k , ground features F_{gp}^k and edge features F_e^k based on shape parameters in each voxel >

2. Scan-To-Model Matching With POU Implicit Surface Representation

<Calculate lidar pose transform T by minimizing the distances between features and POU implicit surface>

for a number of iterations **do**

for each non ground planar feature in F_p^k **do**

Find an POU implicit surface as the correspondence, then compute point to implicit surface distance based on (9) and stack the equation to (12)

for each ground planar feature in F_{gp}^k **do**

Find an POU implicit surface as the correspondence, then compute point to implicit surface distance based on (9) and stack the equation to (12)

for each edge feature in F_e^k **do**

Find an edge line as the correspondence, then compute point to line distance and stack the equation to (12)

Compute a bisquare weight for each row of (12)

Update T for a nonlinear iteration based on Levenberg Marquardt method

if the nonlinear optimization converges **then**

Break;

First, the following equation is used to project feature point x to the map, namely \tilde{x} ,

$$\tilde{x} = Rx + \tau \quad (11)$$

in which, R and τ are the rotation matrix and translation vector corresponding to the pose transform T between current scan and the map. By combining the distances and weights from feature points to their correspondences and equation (11), we derive a geometric relationship between feature points and the corresponding model,

$$f(x, T) = d \quad (12)$$

where each row of f corresponds to a feature point, and d contains the corresponding distances. Finally, we obtain the LiDAR motion with the Levenberg-Marquardt method [33]. We do scan matching between the current scan with the last n scans, and the final result is obtained by aggregating the

successfully matched n_m scans. Literally, by matching the current scan with the historical n scans, the error propagation problem can be suppressed. As for blunder, we limit the scale of the relative transformation between the current scan and the model. When the blunder occurs, we will abandon the result of the current scan and subsequent scan-matching implementation would be slightly affected. After computing the transformation between the scan and model, we add feature voxels corresponding to current scan to the map and remove the feature voxels corresponding to the oldest scan to always keep n scans for scan matching.

V. EXPERIMENTAL EVALUATION*A. Tests On The Public Dataset KITTI*

We evaluate our method on the KITTI odometry benchmark [34], where we use point clouds from a vertical Velodyne HDL-64E S2 mounted on the roof of a car, with a recording rate of 10Hz. The dataset is composed of a wide variety of environments (urban city, rural road, highways, roads with a lot of vegetation, low or high traffic, etc.). The LiDAR measurements are de-skewed with an external odometry, so we do not apply ego-motion algorithm to this dataset. Table I shows the results of our method for all sequences in detail. The relative error is evaluated by the development kit provided with the KITTI benchmark. The data sequences are split into subsequences of 100, 200, ..., 800 frames. The error e_s of each subsequence is computed as:

$$e_s = \frac{\|E_s, C_s\|_2}{l_s} \quad (13)$$

where E_s is the expected position (from ground truth) and C_s is the estimated position of the LiDAR where the last frame of subsequence was taken with respect to the initial position (within given subsequence). The difference is divided by the length l_s of the followed trajectory. The final error value is the average of errors e_s across all the subsequences of all the lengths.

We include here for comparison the reported results of LOAM, a laser-based odometry approach that switches between scan-to-scan and scan-to-model framework, and the reported results of SUMA [16] a laser-based odometry approach using scan-to-model framework. We can see that the proposed method yields results generally on par with the state-of-the-art in laser-based odometry and often achieves better results in terms of translational error. Overall, we achieve an average translational error of 0.61% compared to 0.84% translational error of LOAM on KITTI odometry benchmark. We can see that the proposed method yields results generally on par with the state-of-the-art in laser-based odometry and often achieves better results in terms of translational error.

B. Discussion Of The Parameter And Performance

In our experiment, we define the thresholds c_{th} and p_{th} based on the KITTI Odometry sequence 01. Laser scans contained in this dataset are collected from a highway, the scenario where there is only very few distinct structural features could be used for scan matching. To extract as

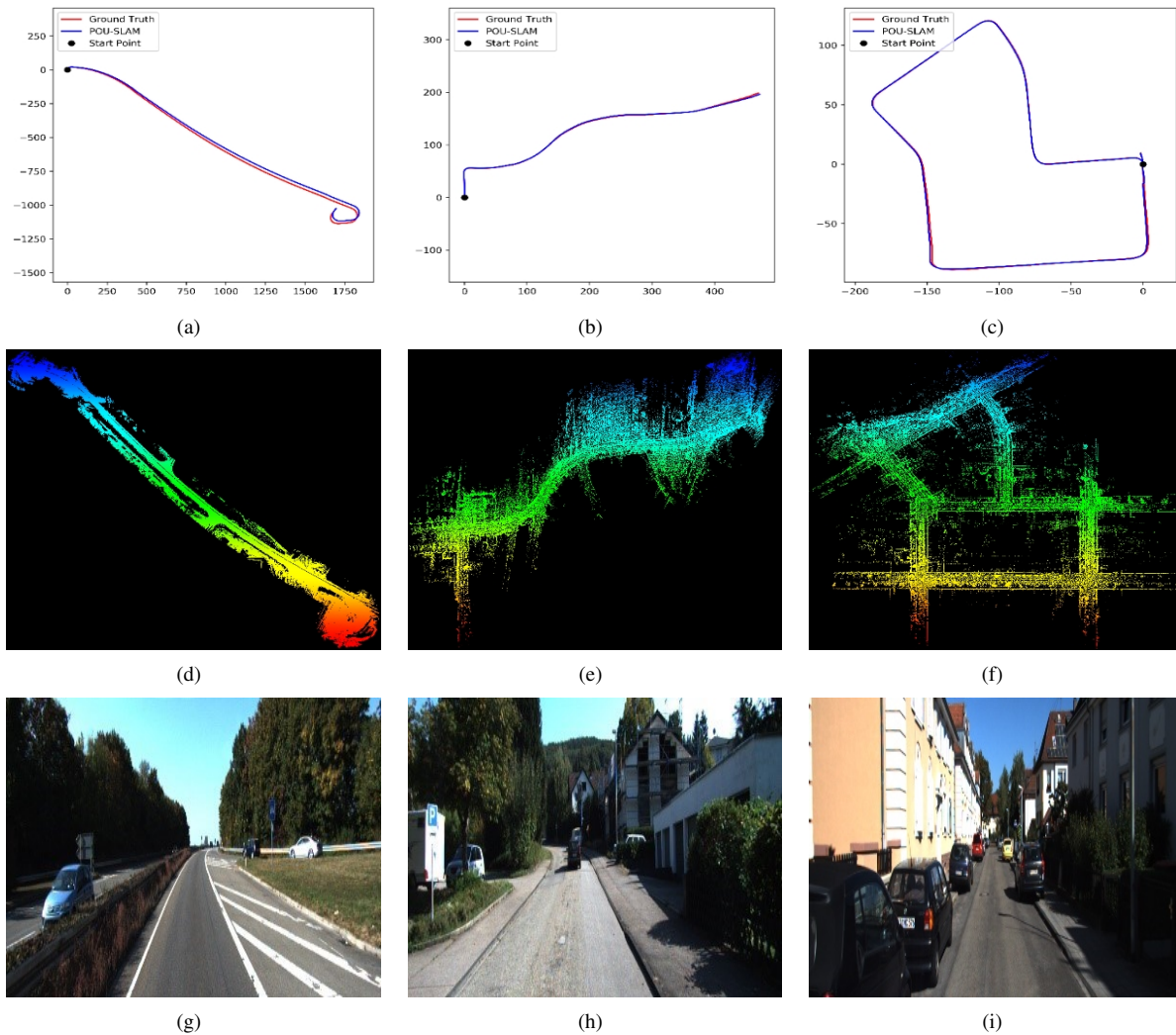


Fig. 4. Sample results using the KITTI odometry benchmark. The datasets are chosen from three types of environments: highway, country and urban from left to right, corresponding to sequences 01, 03, and 07. In (a)–(c), we compare estimated trajectories of the vehicle to the GPS/INS ground truth. The mapping results are shown in (d)–(f). An image is shown from each dataset to illustrate the environment, in (g)–(i).

more as possible features to accomplish the SLAM task, the thresholds should be set to a fairly low level. It is because the smaller the thresholds, features on spheres or cylinders are more likely to be extracted. These sphere and cylinder features themselves are not planar, and if represented by planes directly can bring about errors that would further reduce the accuracy of the downstream algorithms. In this paper, we however blend each planar patch extracted with the weighted function under the implicit surface representation mechanism. This explains why we achieve obvious improvement on this sequence. Therefore, if the thresholds can perform well on such structural feature lacking scenarios, they should be easily transferred to and work in scenarios with richer structural features. For more structured environment such as urban case, the thresholds can be further increased to compromise between the feature richness and the planarity, which promises an accuracy improvement to the whole SLAM algorithm. Table IV shows the influence of c_{th} to the result. When c_{th} is set too large, we can not extract enough features and the result gets worse. Figure 5 shows the variation of the number of

non-ground features in different frames on sequence 01 when c_{th} is set to 0.85. The average number of non-ground features is about 7200. The quantity c as shown in (3) of ground features are generally large and we do not discuss it here. Parameters n_e , n_p and n_{gp} are set to 3 for the KITTI odometry benchmark. Table III shows the influence of parameter n_p . We compare our POU model to the plane model which is fitted as in LOAM. The result of our model is better than plane model and the result reaches the best when n_p is set to 3. When n_p is set too large, the correspondences are far away from the current scan features and do not contribute to local surface reconstruction. The parameter n is set to 40 for the KITTI odometry benchmark. Table II shows the influence of parameter n .

C. Discussion Of The Processing Time

The operating platform in our experiment is Intel i7-7820@3.60 GHz with 16 GB RAM. Our method is evaluated by processing KITTI datasets rather than being deployed in an

autonomous vehicle. With regarding to real time implementation, which heavily depends on the hardware configuration, we cannot assert if our method is real time or not. However, we can still provide some of the parameters so people can know the runtime performance of our method. First, we split the point cloud into voxels which contain certain numbers of points and calculate the linear-ness values and surface-ness value for these voxels. It takes 0.5s. Second, we do scan matching for each feature. The time cost is depending on the number of features. For KITTI Odometry sequence 01, the runtime is about 1.5s. Taking both the above factors into consideration, our SLAM runs at 2s per scan with one thread. For comparison, LOAM runs at 1s per scan on the same KITTI dataset. The reason is that compared to LOAM which extract two-dimensional curvature features, we construct voxel grids and extract features according to the spatial distribution of the points contained in each voxel. This results in better accuracy (refer to Table I) with a slight sacrifice of efficiency.

TABLE I
RESULTS ON KITTI ODOMETRY BENCHMARK

Sequence	Environment	LOAM	SUMA	Our SLAM
0	Urban	0.78%	0.7%	0.64%
1	Highway	1.43%	1.7%	0.90%
2	Urban+Country	0.92%	1.2%	0.74%
3	Country	0.86%	0.7%	0.59%
4	Country	0.71%	0.4%	0.49%
5	Urban	0.57%	0.4%	0.43%
6	Urban	0.65%	0.5%	0.36%
7	Urban	0.63%	0.7%	0.35%
8	Urban+Country	1.12%	1.2%	0.84%
9	Urban+Country	0.77%	0.6%	0.53%
10	Urban+Country	0.79%	0.7%	0.83%

TABLE II
IMPORTANCE OF THE PARAMETER n

Parameter n	Drift on KITTI training dataset
$n=1$	1.34%
$n=5$	0.98%
$n=10$	0.92%
$n=40$	0.90%

TABLE III
COMPARE POU MODEL TO PLANE MODEL

Parameter n_p	Plane Model	POU Model
$n=3$	0.93%	0.90%
$n=4$	0.94%	0.92%
$n=5$	0.94%	0.92%

TABLE IV
THE INFLUENCE OF c_{th} TO THE RESULT

Parameter c_{th}	Drift on KITTI training dataset
$c_{th}=0.95$	1.03%
$c_{th}=0.90$	0.92%
$c_{th}=0.85$	0.90%

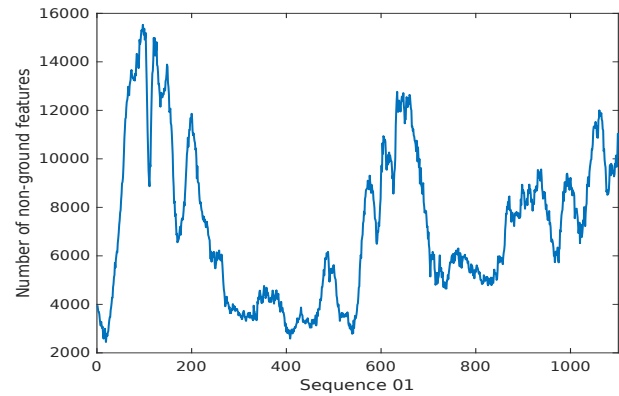


Fig. 5. Variation Of the Number of Non-ground features In Different Frames On Sequence 01

VI. CONCLUSION

We present a new 3D LiDAR SLAM method that is composed of a new feature voxel map and a new scan-to-model matching framework. We build a novel feature voxel map with voxels including salient shape characteristics. In order to adapt to the proposed map, we implement scan-to-model matching using POU implicit surface representation to blend the correspondence voxels in map together. As experimental results illustrate, the proposed method yields accurate results that are on par with the state-of-the-art. Future work will proceed in two directions. From the research perspective, a specific and efficient octree will be designed to get 3D grid. Meanwhile, we will deploy the method to fulfil real time application with the aid of multiple threads or GPU to accelerate data processing.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Fund of China (Grant No. 91848111, 61703387).

REFERENCES

- [1] Jikai Wang and Zonghai Chen. A novel hybrid map based global path planning method. In *2018 3rd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pages 66–70, 2018.
- [2] Jikai Wang, Peng Wang, and Zonghai Chen. A novel qualitative motion model based probabilistic indoor global localization method. *Information Sciences*, 429:284–295, 2018.
- [3] Peng Wang, Qi-Bin Zhang, and Zong-Hai Chen. A grey probability measure set based mobile robot position estimation algorithm. *International Journal of Control Automation and Systems*, 13(4):978–985, 2015.
- [4] Qibin Zhang, Peng Wang, and Zonghai Chen. Mobile robot pose estimation by qualitative scan matching with 2d range scans. *Journal of Intelligent and Fuzzy Systems*, 36(4):3235–3247, 2019.

- [5] Hui Xiong, Youping Chen, Xiaoping Li, Bing Chen, and Jun Zhang. A scan matching simultaneous localization and mapping algorithm based on particle filter. *Industrial Robot-an International Journal*, 43(6):607–616, 2016.
- [6] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems 2014*, volume 10, 2014.
- [7] Ji Zhang and Sanjiv Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41(2):401–416, 2017.
- [8] Michael Bosse and Robert Zlot. Continuous 3d scan-matching with a spinning 2d laser. In *2009 IEEE International Conference on Robotics and Automation*, pages 4244–4251, 2009.
- [9] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050, 2017.
- [10] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.
- [11] Tixiao Shan and Brendan J. Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765, 2018.
- [12] Jean-François Lalonde, Nicolas Vandapel, Daniel F. Huber, and Martial Hebert. Natural terrain classification using three-dimensional lidar data for ground robot mobility. *Journal of Field Robotics*, 23(10):839–861, 2006.
- [13] Yungeun Choe, Inwook Shim, and Myung Jin Chung. Urban structure classification using the 3d normal distribution transform for practical robot applications. *Advanced Robotics*, 27(5):351–371, 2013.
- [14] Haoyang Ye, Yuying Chen, and Ming Liu. Tightly coupled 3d lidar inertial odometry and mapping. *arXiv preprint arXiv:1904.06993*, 2019.
- [15] Michał R. Nowicki, Dominik Belter, Aleksander Kostusiak, Petr Cížek, Jan Faigl, and Piotr Skrzypczyński. An experimental study on feature-based slam for multi-legged robots with rgb-d sensors. *Industrial Robot-an International Journal*, 44(4):428–441, 2017.
- [16] Jens Behley and Cyrill Stachniss. Efficient surfel-based slam using 3d laser range data in urban environments. In *Robotics: Science and Systems XIV*, volume 14, 2018.
- [17] Kejie Qiu and Shaojie Shen. Model-aided monocular visual-inertial state estimation and dense mapping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1783–1789, 2017.
- [18] Frank Moosmann and Christoph Stiller. Velodyne slam. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011.
- [19] Martin Velas, Michal Spanel, and Adam Herout. Collar line segments for fast odometry estimation from velodyne point clouds. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4486–4495, 2016.
- [20] Todor Dimitrov Stoyanov, Martin Magnusson, Henrik Andreasson, and Achim Lilienthal. Fast and accurate scan registration through minimization of the distance between compact 3d ndt representations. *The International Journal of Robotics Research*, 31(12):1377–1393, 2012.
- [21] Jari Saarinen, Todor Stoyanov, Henrik Andreasson, and Achim J. Lilienthal. Fast 3d mapping in highly dynamic environments using normal distributions transform occupancy maps. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4694–4701, 2013.
- [22] François Pomerleau, Francis Colas, and Roland Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104, 2015.
- [23] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.
- [24] Ireneusz Tobor, Patrick Reuter, and Christophe Schlick. Reconstructing multi-scale variational partition of unity implicit surfaces with attributes. *international conference on shape modeling and applications*, 68(1):25–41, 2006.
- [25] Tung-Ying Lee and Shang-Hong Lai. 3d non-rigid registration for mpu implicit surfaces. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2008.
- [26] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942, 2015.
- [27] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.
- [28] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo r-cnn based 3d object detection for autonomous driving. *arXiv preprint arXiv:1902.09738*, 2019.
- [29] Renaud Dube, Daniel Dugas, Elena Stumm, Juan I. Nieto, Roland Siegwart, and Cesar Cadena. Segmatch: Segment based place recognition in 3d point clouds. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5266–5272, 2017.
- [30] Ting Sun, Ming Liu, Haoyang Ye, and Dit-Yan Yeung. Point-cloud-based place recognition using cnn feature extraction. *arXiv preprint arXiv:1810.09631*, 2018.
- [31] Tongtong Chen, Bin Dai, Ruili Wang, and Daxue Liu. Gaussian-process-based real-time ground segmentation for autonomous land vehicles. *Journal of Intelligent and Robotic Systems*, 76(3):563–582, 2014.
- [32] Yutaka Ohtake, Alexander G. Belyaev, Marc Alexa, Greg Turk, and Hans peter Seidel. Multi-level partition of unity implicits. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 463–470, 2003.
- [33] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2000.
- [34] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.