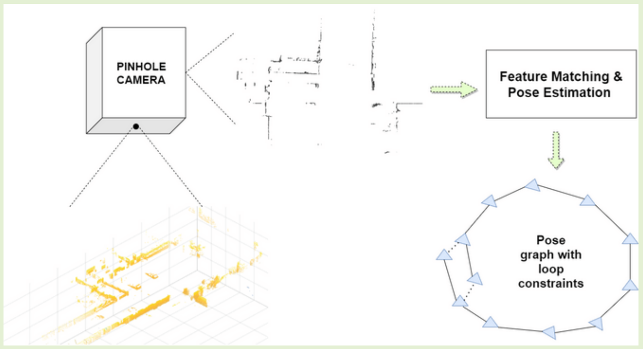
A Feature Based Laser SLAM Using Rasterized Images of 3D Point Cloud

Waqas Ali, Peilin Liu, *Senior Member, IEEE*, Rendong Ying, *Member, IEEE*, and Zheng Gong

Abstract—**An accurate and computationally efficient SLAM algorithm is vital for autonomous vehicles. Most modern SLAM systems use feature detection approaches to limit computational requirements. Feature detection through a 3D point cloud can be a computationally challenging task. In this paper, we propose a feature-based SLAM algorithm using 2D image projections of the 3D laser point cloud. We use a camera parameters matrix to rasterize the 3D point cloud to an image. Then ORB feature detector is applied to these images. The proposed method gives repeatable and stable features in a variety of environments. Based on such features, we can estimate the 6dof pose of the robot. For loop detection, we employ a 2-step approach, i.e., nearest key-frame detection and loop candidate verification by matching features extracted from rasterized LIDAR images. We evaluate the proposed system with implementation on the KITTI dataset. Through experimental results, we show that the algorithm presented in this paper can substantially reduce the computational cost of feature detection from the point cloud and the whole SLAM system while giving accurate results.**


PINHOLE CAMERA
Feature Matching & Pose Estimation
Pose graph with loop constraints

Index Terms—**Laser scanning, place recognition, pose estimation, rasterization, mapping, simultaneous localization and mapping.**

I. Introduction

SLAM [1], [2] problem is associated with the localization of a moving body and building an accurate map of the environment. An accurate and computationally efficient algorithm for simultaneous localization and mapping is integral for modern autonomous vehicles. SLAM system is commonly implemented using two types of sensors, i.e., vision or LIDAR. LIDAR sensors have several advantages over vision-based sensors. It provides highly accurate range data and is not affected by light conditions. By using LIDAR, it is possible to achieve higher accuracy for localization and mapping.

A modern 3D LIDAR sensor is composed of rotating laser beams to acquire 3D range data from the environment. We can divide laser-based simultaneous localization and mapping into two sections based on the implementation approach. The first method involves state estimation by using scan matching of the whole point cloud. This method presents some advantages, i.e., better accuracy and applicability in any environment, but matching a dense point cloud becomes computationally expensive.

Manuscript received July 23, 2021; accepted September 13, 2021. Date of publication September 16, 2021; date of current version October 29, 2021. The associate editor coordinating the review of this article and approving it for publication was Prof. You Li. *(Corresponding author: Waqas Ali.)*

The authors are with the School of Electronic Information and Electrical Engineering, Shanghai Jiaotong University, Shanghai 200240, China (e-mail: vaqas11@sjtu.edu.cn).

Digital Object Identifier 10.1109/JSEN.2021.3113304

The other method is based on scan registration using features extracted directly from the point clouds. Feature-based methods present better computational efficiency. But most of the feature-based methods employed in literature are environment-specific. For instance, corner and plane-based feature detectors are designed for indoor environments [3]–[5] and tree detectors are applied in outdoor settings [6], [7]. These feature detectors are for a specific environment and there has been little work done for general-purpose feature extraction from 3D laser point cloud for SLAM application.

Our system builds on the approach of Li and Olsen [8], i.e., general-purpose feature detection using rasterized LIDAR images. Our main contributions are:

1) A novel SLAM algorithm based on ORB features extracted from the 3D point cloud. Our method can accurately estimate the 6DOF pose and build a map using such features.
2) A lightweight and computationally efficient laser SLAM method because of the fast and efficient feature detection approach.
3) We present a novel loop closure detection technique for the LIDAR-based SLAM system. We search for the nearest key-frames and then match features to detect loop closure.

In our system, we ensure accurate results with the help of multiple levels of optimization. First, the vehicle's pose is estimated and optimized using the least square method. Next, both trajectory and local maps are optimized using local bundle adjustment. Last, we employ loop closure detection for global optimization and to correct the whole trajectory. In this paper, we show the efficiency of the proposed algorithm with the help of experiments on the public dataset. Using the KITTI dataset [9], our method gave results on par with state-of-the-art algorithms at a much lower computational cost.

II. Related Work

A. ICP Methods

For most laser SLAM algorithms, scan matching is used to estimate the relative motion between two scans. ICP algorithms [10], [11] are commonly used for this purpose. It is a simple technique and easy to implement. ICP algorithm starts with data association and then computes transformation to align the two point clouds. We iterate these steps until the solution converges. The ICP algorithm requires a good initial estimate to ensure accurate results. To make ICP more efficient, several variants have been proposed in the literature [12]–[14]. Pomerleau *et al.* [15] proposed a comparison framework between the two most popular ICP algorithms for point cloud registration, i.e., point-to-point and point-to-plane. ICP based methods can produce accurate results depending on a good initial guess. But these algorithms are computationally expensive.

B. Feature Detection

For better computational efficiency, most modern laser SLAM methods extract features from the point cloud to perform scan registration. Dong and Barfoot [16] proposed a method for extracting visual features from intensity and range images acquired from AUTONOSYS LIDAR. Other similar approaches [17], [18] estimate the motion by matching features from images and compute motion model from Gaussian processes. Their method makes intensity and range images using laser and camera, while our method uses a rasterized image from laser point cloud only. Steder *et al.* [19] used point features extracted from panoramic range images for place recognition. But their method still reports higher computational requirements. Zhuang *et al.* [20] introduced a method of transforming point cloud data to a bearing angle image and then extract SURF-based features.

Chong *et al.* [21] proposed a method to form a synthetic 2D LIDAR for the localization of a robot in a 3D environment. Vertical planes are extracted from the 3D point cloud and projected to a virtual plane to form synthetic 2D LIDAR. Then Monte-Carlo localization is applied to this synthetic 2D LIDAR data. A lot of information is lost during the projection of 3D points to 2D planes using such a technique, which affects the result. Zhang and Singh [22] extracted corners and plane features directly from the point cloud and could estimate accurate odometry in real-time. For such a technique, the whole point cloud needs to be scanned for feature detection. While we only need to scan an image to extract features.

Our approach is most similar to the work of Li and Olsen [8]. In their work, they introduced the idea of designing general-purpose feature detectors. They applied a multi-scale

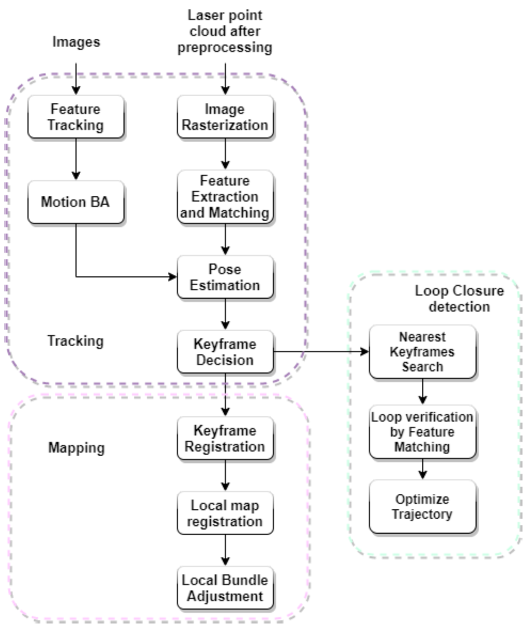


Fig. 1. System overview showing steps involved in tracking, mapping and loop closure threads.

Kanade – Tomasi corner detector on a rasterized image of the point cloud. We take this idea further and use a pinhole camera model for image formation instead of direct rasterization. This permits us to apply advanced feature detectors such as ORB and we can detect stable and repeatable features from such images. We form a gray-scale image by assigning the z-value of each 3D point as the pixel intensity. Using this information we can estimate accurate 3D poses and build a precise map.

C. Loop Closure Detection

For robust SLAM applications, loop closure detection is vital, as it corrects the accumulated error in the estimated trajectory. Hess *et al.* [23] presented the method of dividing the map into sub-maps. They used the branch and bound approach for computing scan to sub-map matches as constraints in the pose graph structure. The optimization is done quickly to ensure operation in real-time. Dubé *et al.* [24] proposed a point cloud segmentation-based approach. An incoming point cloud is divided into segments, its descriptors are extracted and saved into the system. Loop closure candidate is selected by matching the new segments and performing geometric verification. Magnusson *et al.* [25] proposed an approach where global statistics are computed for point cloud and a simple threshold-based approach is used to compute loop closure candidates. Behley and Stachniss [26] proposed a surfel-based mapping approach for building global maps and used a map-based criterion for loop closure detection.

Using ORB features in our approach gives us an edge over existing methods, i.e., loop closure candidate verification based on feature matching is simple and more efficient. In our algorithm, the loop closure detection approach comprises two steps. First, a loop closure candidate is selected based on the nearest key-frame search. Then further verification is

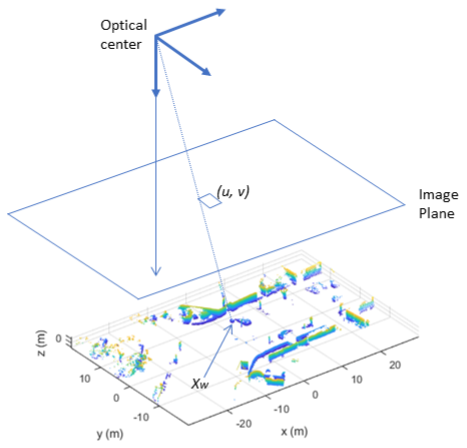


Fig. 2. The process of projecting a 3D point from the LIDAR point cloud onto the image plane. Here z-axis is set as the optical axis.

performed based on feature matching. If these two steps are passed the loop closure constraint is added to the pose-graph structure and the whole trajectory is optimized.

III. Pre-Processing

Our method starts with the pre-processing of the 3D point cloud. We want to extract stable features from the rasterized images. So it is necessary to remove noise and outliers from the raw point cloud. A 3D point cloud recorded by a multi-line LIDAR contains circular rings on the ground plane. If a rasterized image contains these rings, it affects the accuracy of feature detection and motion estimation. We apply Random Sample and Consensus (RANSAC) [27] plane fitting to detect and remove the ground plane. We use the remaining point cloud for image formation.

IV. Rasterization

In our system, we use the camera parameters matrix to project the point cloud to an image plane. Figure 2 shows the projection of 3D points to the image plane. The 3D points in a point cloud are in LIDAR coordinates. We want to project each 3D point to a 2D pixel in image coordinates with intensity value. The image formed because of using the pinhole camera model instead of direct rasterization is suitable for using advanced feature detection methods, such as ORB.

The image rasterization is simple with the point cloud in the LIDAR coordinates. It is first transformed to camera coordinates and then projected to the image plane. For a point in LIDAR coordinates P_l can be transformed to P_c in the camera coordinates using the following equation:

$$P_c = \boldsymbol{R}(\boldsymbol{I}|\boldsymbol{t})P_l \tag{1}$$

Equation1 shows the extrinsic camera parameters, it is given as a rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t}$. Then we define the camera intrinsic parameters, which map a 3D point in the camera coordinates to the image plane. These parameters are given as:

$$\mathbf{K} = \begin{bmatrix} f & 0 & t_u \\ 0 & f & t_v \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The matrix $\mathbf{K}$ defines the intrinsic parameters of a camera. Here f is the focal length and (t_u, t_v) is the optical center of the camera. The final relationship between a point in LIDAR coordinates P_l to a point on image P_i is given as:

$$P_i = \boldsymbol{K}\boldsymbol{R}(\boldsymbol{I}|\boldsymbol{t})P_l = \boldsymbol{C}P_l \tag{3}$$

Using equation 3 we project the point cloud to an image plane to form a gray-scale image. An important step is to save the z-value of each 3D point that gives the rasterized image some physical meaning. We can minimize the loss of information and the re-projection error of feature points. For that purpose, we assign the intensity value of each 2D point on the image plane equal to the z-axis value of its corresponding 3D point. Another issue is that for a point cloud there might be several points existing at the same (x, y) location with different z-values. Since we use the z-axis as the optical axis for projection, so these points project to the same pixel point. In such cases, we use the maximum z-value of these points as the corresponding pixel intensity on the rasterized image. The reason for using the maximum z-value is that it represents the visible height for these points observed by LIDAR.

V. Visual Odometry

There are several state-of-the-art visual odometry approaches present in the literature. We design the visual odometry thread based on the method presented in [28]. As this part is not one of the major contributions of this paper, we provide a brief description of the visual odometry estimation. Visual odometry has two steps, i.e., first features extraction and tracking to estimate camera pose and then perform local bundle adjustment for optimization. We extract FAST corners [29] for each new image, and the detector threshold can be adapted to ensure enough features. Next, ORB descriptors are assigned for each corner [30]. Using feature matching, we can estimate the camera pose. Then, we search for the map points in the last frame based on the motion model. After successfully finding correspondences, we perform local bundle adjustment to optimize the pose. Frame by frame visual odometry poses are fed to the tracking thread, where it is used to optimize the LIDAR pose. The purpose of VO is to ensure that the tracking thread does not fail in low feature environments.

VI. Tracking

A. Feature Extraction

The tracking thread receives rasterized images where the first step is feature extraction. First, Gaussian blur is applied to smooth the rasterized image. Then ORB features [30] are extracted from this image. ORB comprises a oriented FAST corner detector and rotated BRIEF descriptor. We apply the FAST corner detector on the rasterized images and then descriptors are computed for each corner. Figure 3 shows

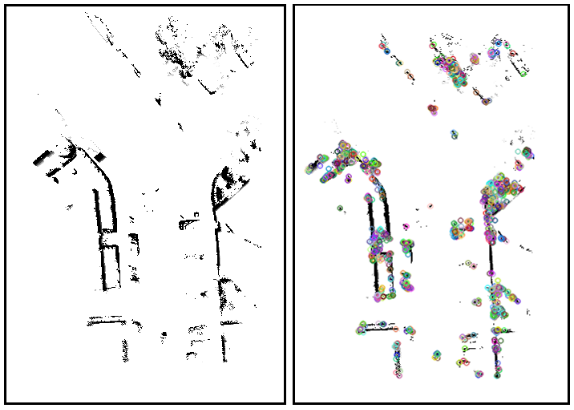


Fig. 3. On the left is an image got after rasterization along the *z*-axis and on the right are ORB features detected from this image.

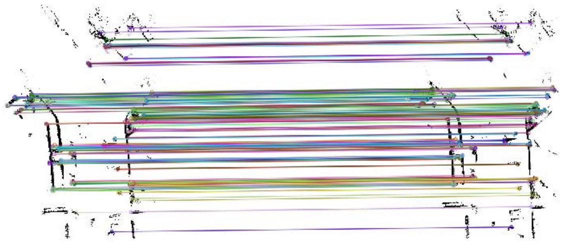


Fig. 4. Example of feature matching from rasterized images.

ORB features detected from a rasterized LIDAR image. Next, we find correspondences between consecutive scans by matching features and remove outliers using the RANSAC algorithm [27]. Figure 4 shows matched features from rasterized images got from 3D LIDAR data. As mentioned in the last section z-values of 3D points are saved during image formation. We use this information and camera parameters to project the matched features to LIDAR coordinates. We then use these features to calculate the LIDAR pose.

B. Pose Estimation and Optimization

The next step is to estimate the motion from the feature points and optimize the pose. If we have a set of feature points extracted from rasterized images projected back to LIDAR coordinates and given as p_t and p_{t-1}. The objective function to estimate motion is defined as follow:

$$f(\boldsymbol{Rot}, \boldsymbol{tr}) = \frac{1}{N_{p_t}} \sum_{i=1}^{N_{p_t}} ||p_{t-1} - \boldsymbol{Rot}\, p_t - \boldsymbol{tr}||^2 \qquad (4)$$

We use the ICP algorithm to estimate the rotation matrix $\boldsymbol{Rot}$ and translation vector $\boldsymbol{tr}$. ICP algorithm can give accurate results depending on a good initial guess for data association. But its main drawback is higher computational cost depending on the number of points. In this paper, we can improve the efficiency of ICP by first using feature points instead of whole point-cloud and second providing correspondences between the points. After estimating the motion between a pair of points, we can calculate the LIDAR pose. At this stage, we build a factor graph for integrating visual odometry into LIDAR pose. We perform a features consistency check and if there are only a few features matched, the LIDAR pose is optimized with VO factors. This vehicle pose is used in the mapping thread to track features and perform local bundle adjustment.

C. Key-Frame Decision

The last step of the tracking thread is to decide when a new key-frame should be selected. We use a simple strategy for key-frame selection based on the work presented in [28]. It has only two requirements i.e.

1) At least 5 frames have passed
2) There are at most 100 points in common with the last key-frame

Whenever a new key-frame K_i is detected, the tracking thread passes this information to the key-frames database where all the key-frames are stored. Mapping and loop-closure threads can access the key-frames information from the database. After the optimization is finished in mapping or loop closure threads we update the key-frames poses using the optimized values. Each key-frame K stores the following information:

1) 6DOF pose
2) 3D position of the feature points seen at that position
3) Descriptors of these feature points

VII. Mapping

A. Map-Points Registration

The mapping thread starts by building a map containing key-frames and map-points as nodes. When a new key-frame K_i is inserted into the database by the tracking thread, the mapping thread adds its pose to the graph as a node. The feature points seen at that key-frame are first tracked using the key-frame's pose and then used to build a local map. At the very first key-frame insertion K_0, the local map is initialized using all the points seen at key-frame K_0. When the next key-frame K_1 comes, we match its feature points against the local map and add the unmatched points into the local map. We repeat these steps with every new key-frame. The key-frame poses and map-points in the local map are optimized using local bundle adjustment.

The bundle adjustment is only for local optimization, so only a set of current key-frames are kept in the graph structure. It requires a culling strategy for key-frames and map-points in the local map. The culling scheme for key-frames in our algorithm is kept simple. During local bundle adjustment, we only optimize current n key-frames. So when we receive a new key-frame, redundant key-frames are removed from the graph. We also need to perform map-points culling. After removing the redundant key-frames, we check for unnecessary map-points. Any map-point in the local map that is not visible by any of the current key-frames is removed. The two culling techniques ensure that our algorithm remains lightweight and computationally efficient.

B. Bundle Adjustment

We use local bundle adjustment to optimize n current key-frames and map-points inside the local map. The map-points are projected to LIDAR coordinates from world coordinates and then to camera coordinates. By solving the local BA problem, we want to minimize the re-projection error of the 3D positions of map-points. The key-frame pose K_{iw} is the LIDAR pose registered as a node in the graph. The transformation from LIDAR to camera coordinates remains fixed. For a point observed in an image $\mathbf{x}_{ij}$ and $\hat{\mathbf{x}}_{ij}$ is computed by re-projecting the 3D point, the cost function to minimize re-projection error is given as:

$$e_{ij} = \mathbf{x}_{ij} - \hat{\mathbf{x}}_{ij}(\mathbf{C}, \mathbf{K}_{iw}) \tag{5}$$

The cost function is for a key-frame node i connected through a constraint to a map-point node j. $\boldsymbol{C}$ represents a function to project a 3D point to an image as derived in equation 3. It includes the intrinsic and extrinsic camera parameters. Note that the camera projection function $\boldsymbol{C}$ defined in section-5 remains constant. Our goal is to minimize equation 5 to find the optimal configuration for nodes.

$$F(x^*) = argmin \sum_{ij} e_{ij}^T \boldsymbol{\Omega}_{ij} e_{ij} \tag{6}$$

Where $\boldsymbol{\Omega_{ij}}$ is the covariance matrix associated with the observation. We define the Jacobian $\boldsymbol{J}_{ij}$ to solve equation 6 as follows:

$$\boldsymbol{J}_{ij} = \begin{pmatrix} 0 \ldots 0 & \boldsymbol{A}_{ij} & 0 \ldots 0 & \boldsymbol{B}_{ij} & 0 \ldots 0 \end{pmatrix}$$
$$\boldsymbol{A}_{ij} = \frac{\partial e_{ij}}{\partial x_i}, \quad \boldsymbol{B}_{ij} = \frac{\partial e_{ij}}{\partial x_j}$$

We find the derivative of the cost function in equation 5 for node i and j to estimate the terms in Jacobian. For combining the camera projection parameters and LIDAR pose to get the relationship between a point in the image and the world coordinates, we get the following expression

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & t_u \\ 0 & f & t_v \\ 0 & 0 & 1 \end{bmatrix} [\boldsymbol{R} \quad \boldsymbol{t}] [\boldsymbol{R}_{iw} \quad \boldsymbol{t}_{iw}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{7}$$

For our system $\boldsymbol{R}$ and $\boldsymbol{t}$ are camera extrinsic parameters, $\boldsymbol{R}$ is used as identity and $\boldsymbol{t}$ is constant. $\boldsymbol{R}_{iw}$ and $\boldsymbol{t}_{iw}$ are the rotation matrix and translation vector of LIDAR pose respectively. f is the focal length of the camera and (t_u, t_v) is the optical center of the camera. The left-hand side of equation 7 represents the point (u, v) on an image written in homogeneous coordinates. In equation 7 the terms to be optimized are X_w, Y_w, Z_w and $\boldsymbol{R}_{iw}$ and $\boldsymbol{t}_{iw}$. The 2 terms of Jacobian are given as follows:

$$\boldsymbol{A}_{ij} = \begin{bmatrix} \dfrac{X}{Z} & 1 & 0 \\ \dfrac{Y}{Z} & 0 & 1 \end{bmatrix} \tag{8}$$

$$\boldsymbol{B}_{ij} = \begin{bmatrix} \dfrac{f}{Z} & 0 & -\dfrac{fX}{Z^2} \\ 0 & \dfrac{f}{Z} & -\dfrac{fY}{Z^2} \end{bmatrix} \tag{9}$$

Once the local BA is complete, we update the key-frame poses and the optimized map-points are saved to the map.

VIII. Loop-Closure Detection

The loop closure detection for our system has two steps, i.e., candidate selection by searching for nearest key-frames and then use feature matching for candidate verification. Using ORB features makes the proposed method fast and efficient. The purpose of loop closure detection is global optimization to correct drift in trajectory. Loop closure thread starts by building a pose-graph that only comprises key-frames poses as nodes. As we receive the new key-frame information from tracking, it is added as a node to the graph. The constraints between these nodes are the estimated 6DOF transformation from the tracking thread.

Although we are using a vision-based feature detection technique for rasterized images, these images and features in our system differ from the images of an actual camera. So we design an approach that is applicable for LIDAR rasterized images and find the loop closure accurately. When the tracking adds a new key-frame K_i to the database, we search for the nearest key-frames using a distance threshold. If key-frame K_l falls inside the distance threshold of key-frame K_i, we match features seen at K_i and K_l. First, we apply the ratio test [31] to check for outliers. If there are enough features after the ratio test, we use RANSAC to check for more outliers and verify the selected candidate. Once we have found a loop closure candidate with final verification, we estimate a 6DOF transformation between K_i and K_l. We use the estimated transformation to add a loop constraint between the respective nodes in the graph. We use the Levenberg Marquardt optimizer of the GTSAM [32] library for the optimization. After the optimization is complete, we update all the poses in the trajectory.

IX. Experiments

We have implemented our algorithm on the KITTI dataset [9] to evaluate the performance. We run the tests on a laptop with an i5-8300H processor and 8GB RAM. To validate the performance, we use the open-source implementations of LOAM [22] and cartographer [23]. KITTI odometry benchmark dataset provides 22 sequences of image and LIDAR data. Sequences 0-10 are training sequences with ground truth, and sequences 11-21 are for testing system performance. They collect data from three types of environments, i.e., country, highway, and urban. Cartographer [23] requires IMU data besides laser scans for the 3D SLAM. So, we use corresponding sequences from KITTI raw datasets for cartographer experiments. We use three criteria for the performance assessment, i.e., tracking accuracy, loop closure performance, and computational costs.

A. Tracking Performance

One of the most important goals of any SLAM system is to ensure tracking accuracy. We use two state-of-the-art methods LOAM [22] and cartographer [23] for localization accuracy comparison. Absolute trajectory error [33] is used to evaluate the performance of a complete SLAM system along with loop closure. We align the trajectory computed by our system with the ground truth, then estimate absolute error. We analyze the

TABLE I
ABSOLUTE TRAJECTORY ERROR VALUES OF OUR SYSTEM AGAINST LOAM

Sequence	Method	RMSE	SD
00	Our method	**7.6581**	**3.1464**
	LOAM	13.8885	6.3744
01	Our method	**21.0776**	**9.5753**
	LOAM	47.9101	29.4392
02	Our method	**16.6212**	9.3645
	LOAM	19.8842	**5.8554**
03	Our method	**1.6546**	**0.7379**
	LOAM	3.6803	2.0770
04	Our method	**0.9366**	**0.4709**
	LOAM	2.7308	1.4745
05	Our method	4.4767	2.5246
	LOAM	**4.3881**	**1.9872**
06	Our method	**3.5110**	**1.3098**
	LOAM	3.6839	1.9197
07	Our method	3.5085	1.6567
	LOAM	**1.8219**	**0.6903**
08	Our method	**11.6740**	**2.1580**
	LOAM	15.0168	6.8326
09	Our method	**6.3078**	**2.7865**
	LOAM	7.9374	3.0628
10	Our method	**5.2820**	**2.9543**
	LOAM	7.1821	3.6101

TABLE II
RMSE (ROOT MEAN SQUARE ERROR) AND STD (STANDARD DEVIATION) ARE USED TO EVALUATE THE PERFORMANCE OF OUR SYSTEM AGAINST CARTOGRAPHER

Seqeunce	Method	RMSE	SD
2011 10 03 drive 0027	Our method	**7.6581**	**3.1464**
	Cartographer	11.6690	4.4771
2011 09 30 drive 0018	Our method	**4.4767**	**2.5246**
	Cartographer	15.2227	6.0145
2011 09 30 drive 0020	Our method	**3.5110**	**1.3098**
	Cartographer	25.3995	5.103
2011 09 30 drive 0027	Our method	**3.5085**	**1.6567**
	Cartographer	5.0272	3.3951
2011 09 30 drive 0033	Our method	**6.3078**	**2.7865**
	Cartographer	13.1488	6.5532

performance through the standard deviation and root mean square error (RMSE) of the absolute trajectory error.

First, we compare the performance of our system with LOAM. LOAM is also a feature-based method that extracts corner and plane features from a 3D point cloud. We run the open-source implementation of LOAM on the 11 training sequences of the KITTI dataset. Table I shows the standard deviation and root mean square values of the 11 sequences for our method and LOAM. Our system's overall performance is better as compared to LOAM for the training dataset. Our method's performance is much better in longer sequences such as 00 and 02 due to loop closure optimization. The trajectory of LOAM drifts away from ground truth for longer sequences, but our method gives better accuracy.

TABLE III
A COMPARISON OF THE RESULTS OF GLOBAL BA AND POSE GRAPH OPTIMIZATION WITHOUT LANDMARKS FOR LOOP CLOSURE OPTIMIZATION

	Global BA	Pose graph optimization
RMSE±SD	7.4172±2.6985m	7.6581±3.14m
Computational Time	15.731s	0.555s
CPU Load	26.72±9.14%	13.41±6.51%

Next, we compare the tracking performance with the cartographer. Both systems contain loop closure detection. So we select the sequences with loop closure to compare the performance. Table II shows the ATE values for the five sequences. Our method performs better for all the sequences because to superior loop closure performance.

We submitted the results of the 11 test sequences to the KITTI odometry evaluation benchmark. The overall translation error for our method is 1.47% and the rotation error is 0.0033 deg/m. We found that the main reason for lower translation accuracy on test sequences was fast-moving dynamic objects such as cars in sequences along the highway.

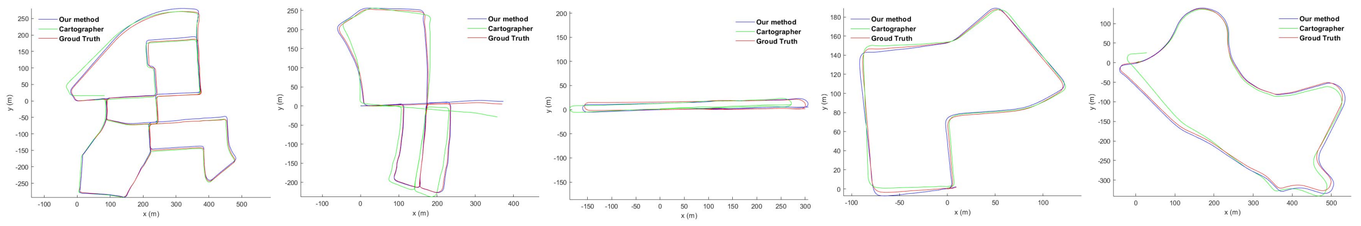
B. Global Bundle Adjustment

We implemented global bundle adjustment for loop closure optimization for comparison with the method used in this paper. Table III shows the results for sequence 00 of the KITTI dataset. We used three parameters to assess the performance of two methods, i.e., RMSE error, computational time, and CPU load during optimization. The trajectory error for the global BA is slightly lower than our method. But, it reports much higher computational time and CPU load. The reason for high computational requirements is that for global BA we have to optimize both keyframe poses and all the map-points. For this reason, we preferred using pose graph optimization without map-points. As it gives accurate results at a much lower computational cost.

C. Loop Closure Performance

One of the major contributions of this paper is a novel loop closure method for laser SLAM. Our approach is simple, i.e., find nearest key-frames and loop verification based on feature matching. LOAM comprises only tracking and mapping threads, so we use the cartographer to evaluate the efficiency of our loop closure method. Figure 5 shows the final trajectories of our method and cartographer for the five KITTI dataset sequences. Sequences 2011_09_30_drive_0027 and 2011_09_30_drive_0033 are important for comparison. Because in these two sequences, there are only a few frames to detect loop closure at the end of trajectories. In this scenario, the cartographer lagged and could not detect loop closure. Our systems detected loop closure efficiently for both cases.

Sequence 2011_10_03_drive_0027 is the longest sequence with 4550 frames and contains several loop closures. The cartographer can detect most of the loops correctly but



(a) 2011_10_03_drive_0027 (b) 2011_09_30_drive_0018 (c) 2011_09_30_drive_0020 (d) 2011_09_30_drive_0027 (e) 2011_09_30_drive_0033

Fig. 5. Trajectory plots after loop closure detection and optimization for KITTI sequences.

TABLE IV
MEANTIME AND ITS STANDARD DEVIATION ESTIMATED FOR IMAGE FORMATION, TRACKING AND MAPPING THREADS IN OUR SYSTEM FOR SEQUENCE 00 OF KITTI DATASET

Operation	Mean Time (ms)	SD (ms)
Image Projection	6	3.6
Feature Extraction and matching	32.5	13.1
Pose Estimation	3.2	0.98
Key-frame and map-points registration	5.8	2.5
Local BA	55.7	11.2

drifts at the end because of failing to detect loop closure. The same problem occurs in the result of the sequence 2011_09_30_drive_0018. Our system can find loop closure accurately for both sequences. Both methods showed good performance for sequence 2011_09_30_drive_0020, which contains loop closure with several frames at the end of the trajectory. The proposed technique for loop detection is fast and precise. We eliminate false matches by ratio test and RANSAC by removing outliers. The verification process requires less time and can give accurate matches. We successfully prove the capability of our method from these results.

D. Computational Efficiency

The primary goal of this paper is to design a lightweight SLAM algorithm. We compared the localization accuracy with LOAM and cartographer. In this section, we have a look at the computational requirements of the three systems. Table IV presents the values of the time required for each section of our system. The tracking thread has three parts, i.e., Image projection, Feature extraction and matching and Pose estimation. The mean time spent on tracking during KITTI dataset experiments is around 41.7 ms. The mapping thread takes a mean time of 61.5 ms to complete key-frames and map-points registration and performs local bundle adjustment.

Figure 6 and figure 7 show a comparison of the meantime and standard deviation for tracking and mapping threads for our system and LOAM. Our system performs these tasks in much less time as compared to LOAM. Tracking has less time consumption because to the fast feature extraction method. LOAM algorithm extracts corner and planner features by scanning through the 3D point cloud. But for our system, we only need to scan a rasterized image at each frame to extract feature points, which substantially reduced the time requirement of the complete system.

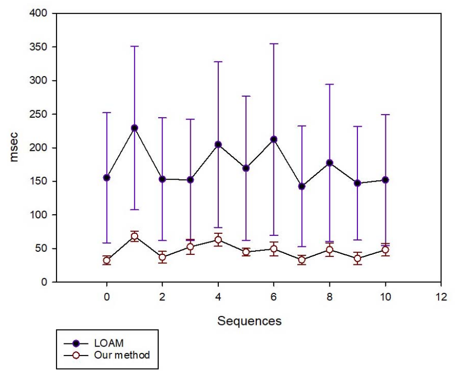


Fig. 6. A comparison of time consumed by the Tracking thread for our method and LOAM.

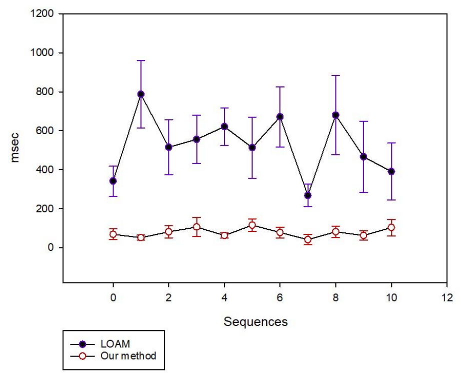


Fig. 7. A comparison of time consumed by the Mapping thread for our method and LOAM.

Next, we evaluate the computational efficiency of our method with comparison to the cartographer. Both systems comprise local and global parts, local thread deals with tracking and map registration. While the global thread handles loop closure detection and global optimization. First, we look at the time requirements of both methods for local SLAM during experiments on the KITTI dataset. Figure 8 shows the mean time required for local SLAM calculations along with standard deviation. Our method takes much less time across all experiments because we use features for pose estimation while the cartographer relies on direct scan matching.

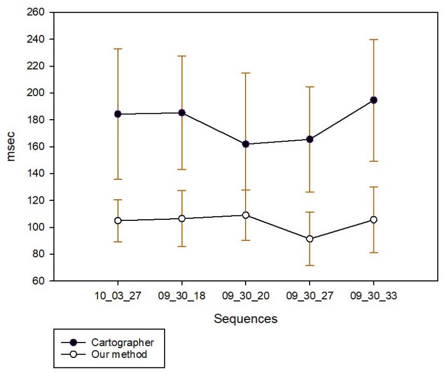


Fig. 8. A comparison of the time consumed by local SLAM for our method and cartographer.

TABLE V
MEAN PERCENTAGE VALUES OF CPU USAGE BY CARTOGRAPHER AND OUR METHOD

	Cartographer	Our method
10_03_27	32.62±9.15	**13.41±6.51**
09_30_18	33.6±8.25	**8.51±3.54**
09_30_20	26.55±11.94	**5.35±1.58**
09_30_27	18.59±4.78	**5.16±1.74**
09_30_33	23.03±5.7674	**6.58±3.12**

We can see the comparison of the CPU load for both systems in Table V. The values from the table show resources required by each system to run an online SLAM system including local estimation and global optimization based on loop closure search. It is clear from the results that the cartographer requires more resources to run the complete SLAM algorithm.

E. Discussion

From these experiments, we have been able to validate the performance of the proposed system. We designed a light-weight SLAM algorithm that can produce accurate localization on par with state-of-the-art methods, with superior loop closure detection. In this paper, we used visual odometry to assist in low feature environments. A low-resolution LIDAR such as a 16 line or 32 lines sensor can also be used for the proposed method. Because even with lower resolution the structural information is present on the rasterized image and the system performance is not affected. However, there are some limitations to the proposed approach. The presence of fast-moving dynamic objects causes outliers in the features detected from the rasterized images. These outliers can cause a drop in accuracy. The feature detection in open spaces or structureless environments can be challenging. The addition of other sensors and re-localization can be helpful to improve accuracy in such conditions.

X. CONCLUSION

In this paper, we presented a SLAM method based on ORB features extracted from laser rasterized images. We estimate the 6DOF pose and register map using these features. We also designed a loop closure detection method using these features. We have validated the performance of our system through implementation on KITTI datasets. We showed that our system produced accurate results with a substantial reduction in computational cost. One issue seen in experiments is that dynamic objects in the environments can affect the system's performance. Our next research goal is to design re-localization and an efficient mapping strategy to deal with static and dynamic objects in the environment.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics* (Intelligent Robotics and Autonomous Agents). Cambridge, MA, USA: MIT Press, 2005.

[2] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, 2010.

[3] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2D range data for indoor mobile robotics," *Auto. Robots*, vol. 23, no. 2, pp. 97–111, Aug. 2007.

[4] E. B. Olson, "Robust and efficient robotic mapping," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 2008.

[5] A. Diosi, L. Kleeman, A. Diosi, and L. Kleeman, "Uncertainty of line segments extracted from static sick PLS laser scans," in *Proc. Austral. Conf. Robot. Autom.*, 2003, pp. 1–10.

[6] M. Rutzinger, B. Höfle, M. Hollaus, and N. Pfeifer, "Object-based point cloud analysis of full-waveform airborne laser scanning data for urban vegetation classification," *Sensors*, vol. 8, no. 8, pp. 4505–4528, Aug. 2008.

[7] M. Voss and R. Sugumaran, "Seasonal effect on tree species classification in an urban environment using hyperspectral data, LiDAR, and an object- oriented approach," *Sensors*, vol. 8, no. 5, pp. 3020–3036, May 2008.

[8] Y. Li and E. B. Olson, "Extracting general-purpose features from LIDAR data," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 1388–1393.

[9] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[10] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Proc. SPIE*, vol. 1611, Apr. 1992, pp. 586–606.

[11] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," *Image Vis. Comput.*, vol. 10, no. 3, pp. 145–155, 1992.

[12] A. Censi, "An ICP variant using a point-to-line metric," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 19–25.

[13] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3D-NDT," *J. Field Robot.*, vol. 24, no. 10, pp. 803–827, 2007.

[14] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2D range scans," *J. Intell. Robot. Syst.*, vol. 18, no. 3, pp. 249–275, 1997.

[15] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Auto. Robots*, vol. 34, no. 3, pp. 133–148, Apr. 2013.

[16] H. Dong and T. Barfoot, "Lighting-invariant visual odometry using lidar intensity imagery and pose interpolation," in *Proc. 7th Int. Conf. Field Service Robots*, Matsushima, Japan, Jul. 2012.

[17] S. Anderson and T. D. Barfoot, "RANSAC for motion-distorted 3D visual sensors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 2093–2099.

[18] C. H. Tong and T. D. Barfoot, "Gaussian process Gauss–Newton for 3D laser-based visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 5204–5211.

[19] B. Steder, G. Grisetti, and W. Burgard, "Robust place recognition for 3D range data based on point features," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 1400–1405.

[20] Y. Zhuang, N. Jiang, H. Hu, and F. Yan, "3-D-laser-based scene measurement and place recognition for mobile robots in dynamic indoor environments," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 2, pp. 438–450, Feb. 2013.

[21] Z. J. Chong, B. Qin, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus, "Mapping with synthetic 2D LIDAR in 3D urban environment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 4715–4720.
[22] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot., Sci. Syst. (RSS)*, vol. 2, Rome, Italy, 2014, p. 9.
[23] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1271–1278.
[24] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "SegMatch: Segment based loop-closure for 3D point clouds," 2016, *arXiv:1609.07720*. [Online]. Available: http://arxiv.org/abs/1609.07720
[25] M. Magnusson, H. Andreasson, A. Nuchter, and A. J. Lilienthal, "Appearance-based loop detection from 3D laser data using the normal distributions transform," in *Proc. IEEE Int. Conf. Robot. Autom.* May 2009, pp. 23–28.
[26] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," in *Proc. Robot., Sci. Syst. (RSS)*, Pittsburgh, PA, USA, 2018.
[27] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[28] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[29] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2006, pp. 430–443.
[30] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. ICCV*, vol. 11, no. 1. Nov. 2011, pp. 2564–2571.
[31] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.
[32] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Inst. Technol., Atlanta, GA, USA, Tech. Rep. GT-RIM-CP&R2012-002, Sep. 2012.
[33] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D slam systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580.

Waqas Ali is pursuing the Ph.D. degree with the Department of Information and Communication Engineering, Shanghai Jiaotong University. His main areas of research interests are laser SLAM and autonomous navigation for ground and aerial vehicles.

Peilin Liu (Senior Member, IEEE) received the Doctor of Engineering degree from The University of Tokyo, Tokyo, Japan, in 1998. In 1999, she worked as a Researcher at The University of Tokyo. From 1999 to 2003, she was a Senior Researcher at the Central Research Institute of Fujitsu, Tokyo, where her research topics include multimedia processing, IC design, and high-performance processor architecture. She joined Shanghai Jiao Tong University, China, in 2003, where she is currently a Professor with the Department of Electronic Engineering.

Rendong Ying (Member, IEEE) received the B.S. degree from East China Normal University, Shanghai, China, in 1994, and the master's and Ph.D. degrees in electronic engineering from Shanghai Jiao Tong University in 2001 and 2007, respectively. He is now an Associate Professor at the Department of EE, Shanghai Jiaotong University. He is the author of *Embedded System-Principle and Design* (Publishing House of Electronics Industry, 2011). His research areas include digital signal processing, SoC architecture, and machine thinking.

Zheng Gong was born in Beijing, China. He received the B.Eng. (Hons.) degree in electronic and computer engineering and the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 2014 and 2021, respectively. His main areas of research interest are all source positioning and navigation (ASPN) and C-V2X.