2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Automatic Calibration of Multiple 3D LiDARs in Urban Environments

Jianhao Jiao, Yang Yu, Qinghai Liao, Haoyang Ye, Rui Fan, Ming Liu

***Abstract*— Multiple LiDARs have progressively emerged on autonomous vehicles for rendering a rich view and dense measurements. However, the lack of precise calibration negatively affects their potential applications. In this paper, we propose a novel system that enables automatic multi-LiDAR calibration method without any calibration target, prior environment information, and manual initialization. Our approach starts with a hand-eye calibration by aligning the motion of each sensor. The initial results are then refined by an appearance-based method by minimizing a cost function constructed by point-plane distance. Experimental results on simulated and real-world data demonstrate the reliability and accuracy of our calibration approach. The proposed approach can calibrate a multi-LiDAR system with the rotation and translation errors less than 0.04rad and 0.1m respectively for a mobile platform.**

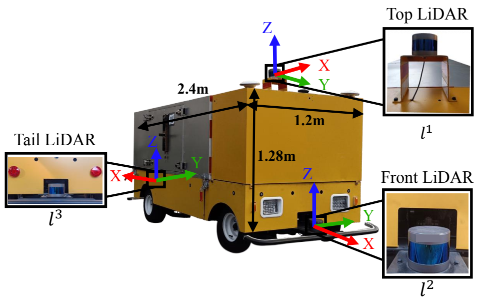
I. Introduction

Accurate extrinsic calibration has become increasingly essential for the broad applications of multiple sensors. Numerous research work has been studied on [1]–[3]. Over the past decades, LiDARs have appeared as a dominant sensor in mobile robotics for their active nature of providing accurate and stable distance measurements. They have been widely utilized in mapping [4] and object detection [5]. However, LiDARs do not provide a high spatial resolution of measurements and are also sensitive to occlusion. These drawbacks limit their potential applications in robotic systems. Fig. 1 (bottom) displays two examples, which is a point cloud captured by the top LiDAR. In block A, the pedestrians and vehicles are scanned with a few points, making the detection of these objects challenging. About block B, points are gathering together because of occlusion. Therefore, employing the multi-LiDAR setup on self-driving cars is necessary.

Traditional calibration techniques for multiple sensors are done by either placing markers in scenes or hand-labeled correspondences. However, these approaches suffer from impracticality and limited scalability to the multi-LiDAR configuration. Additionally, there are surprisingly few discussions on calibrating multiple 3D LiDARs. The majority of current approaches involve one or more of the following assumptions: prior knowledge about the structure of environments [6], usage of additional sensors [7], and user-provided initial values of the extrinsic parameters [8]. It is also not trivial to register point clouds accurately without initial guess. Inspired by the progress of the hand-eye calibration, we find that the extrinsic parameters can be recovered from individual motion provided by each LiDAR. Moreover, the geometric features in environments also form constraints to sensors' relative transformations. Hence, we can conclude that the complementary usage of these approaches is a prospective solution to calibrate multiple LiDARs,

This work was supported by the National Natural Science Foundation of China (Grant No. U1713211), the Research Grant Council of Hong Kong SAR Government, China, under Project No. 11210017, and No. 21202816, awarded to Prof. Ming Liu.

Jianhao Jiao, Yang Yu, Qinghai Liao, Haoyang Ye, Rui Fan and Ming Liu are with RAM-LAB `https://ram-lab.com/`, in the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong SAR, China. {`jjiao, yang.yu, qinghai.liao, hy.ye, eeruifan, eelium`}@ust.hk



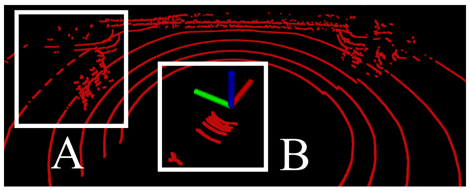


Fig. 1. (Top) Our vehicle consists of a multi-LiDAR system with unknown extrinsic parameters. (Bottom) A point cloud captured by l^1. The white boxes indicate two drawbacks presented in a single LiDAR configuration: (A) measurement sparsity and (B) occlusion.

In this paper, we proposed a novel system[1] which allows automatic calibration of multiple LiDARs in urban environments. It consists of three components: **motion estimation** of each sensor, motion-based **initialization**, and appearance-based **refinement**. We show a variety of experiments to demonstrate the reliability and accuracy of this approach. The contributions of this paper are summarized as follows:

- A pipeline to automatically calibrate the extrinsic parameters of multiple LiDARs which releases the assumptions of calibration targets, prior knowledge about surroundings, and initial values given by users.
- The usage of the motion-based method for initialization and appearance-based method for refinement.
- Extensive experiments on simulated and real-world data.

The rest of the paper is organized into the following sections. In Sect. II, the relevant literature is discussed. An overview of the system pipeline is given in Sect. III. The methodology of our approach, which includes motion estimation, automatic initialization, and refinement is introduced in Sect. IV, followed by experimental results presented in

[1]Code is available at `https://github.com/ram-lab/MLC`.

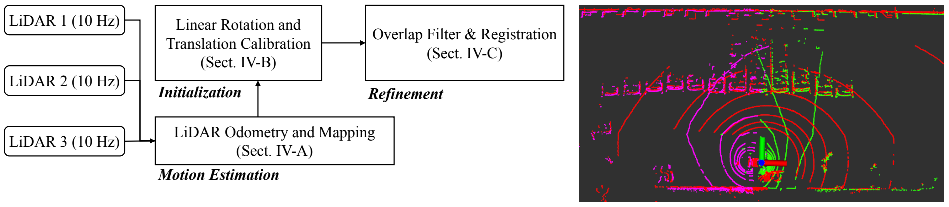

LiDAR 1 (10 Hz)
LiDAR 2 (10 Hz)
LiDAR 3 (10 Hz)
Linear Rotation and Translation Calibration (Sect. IV-B)
Initialization
Overlap Filter & Registration (Sect. IV-C)
Refinement
LiDAR Odometry and Mapping (Sect. IV-A)
Motion Estimation

Fig. 2. This figure illustrates the full pipeline of the proposed approach (left) and the fused point clouds after calibration (right). Note that the red, green, and purple point clouds are captured by the top LiDAR, front LiDAR, and tail LiDAR respectively.

Sect. V. Finally, Sect. VI summarizes the paper and discusses possible future work.

II. RELATED WORK

Besides to multi-LiDAR calibration, there are also extensive discussions on the calibration among LiDARs, cameras, and IMUs. In this section, we categorize them as appearance-based and motion-based methods.

A. Appearance-based approaches

Appearance-based approaches that recover the spatial offset using appearance cues in surroundings are considered as a category of registration problem. The key challenge is searching correspondences among data. Artificial markers that are observable to sensors have been prevalently used to acquire correspondences. Our previous work [6] calibrated the same sensors using three planar surfaces. For calibrating the LiDAR-camera setup, Zhou et al. [3] demonstrated a technique to form line and plane constraints between the two sensors in the presence of a chessboard, while Liao et al. [9] published a toolkit using an arbitrary polygon, which is more general. However, all these methods require markers in scenes. Our approach only utilizes common features such as edges and planes in outdoor environments.

The automatic markerless calibration in arbitrary scenes has led the trend recently. He et al. [7] extracted geometric features among scan points to achieve robust registration, and their work was extended to a challenging scenario [10]. Levinson [8] first put forward an online calibration for a camera-LiDAR system. This is accomplished by aligning edge points with image contours and minimizing a cost function. However, their success highly relies on the initialization given by users. Compared with them, our approach can roughly recover the extrinsics from the sensor's motion, which enables calibration without human intervention.

B. Motion-based approaches

The motion-based approaches treat calibration as a well-researched hand-eye calibration problem [11], where the extrinsic parameters are computed by combining the motions of all available sensors. The hand-eye calibration problem is usually referred to solve $\mathbf{X}$ in $\mathbf{AX} = \mathbf{XB}$, where $\mathbf{A}$ and $\mathbf{B}$ are the motions of two sensors, and $\mathbf{X}$ is their relative transformation. As described in [12], this problem has been addressed since the 1980s. The ongoing research focuses on the calibration of multiple sensors in outdoor environments. Heng et al. [1] proposed CamOdoCal, a versatile algorithm with a bundle adjustment to calibrate four cameras. Taylor et al. [2] provided a more general solution to multi-modal sensors. As presented in [13], motion-based approaches can also be utilized to estimate temporal offset between sensors. Furthermore, several state-of-the-art visual-inertial navigation systems adopted the motion-based approaches for online calibration [14]. Although motion-based methods have been extensively developed, their accuracy is sensitive to the accumulated drifts of estimated motion. In contrast, our method takes advantages of geometric features to improve the calibration of multiple LiDARs.

III. OVERVIEW

The notations are defined as follows. We denote $[0, K]$ the time interval during calibration, and define $\{l_k^i\}$ as the sensor coordinate system of the i^{th} LiDAR at timestamp k. The $x-$, $y-$ and $z-$ axes of coordinate systems are pointing forward, left and upward respectively. We denote I the number of LiDARs to be calibrated, and l^1 the reference LiDAR. The transformation, rotation, and translation from $\{a\}$ to $\{b\}$ are denoted by $\mathbf{T}_b^a$, $\mathbf{R}_b^a$, and $\mathbf{t}_b^a$ respectively. $\mathbf{T}_{l^i}^{l^1}$ or $(\mathbf{R}_{l^i}^{l^1}, \mathbf{t}_{l^i}^{l^1})$ are the unknown transformations from $\{l^1\}$ to $\{l^i\}$ to be solved. LiDARs are synchronized, where point clouds are captured at the same time. We assume that the vehicle is able to perform sufficient motions over a planar surface. With these notations and assumptions, the calibration problem can be defined as:

Problem: Given a sequence of point clouds during calibration, computing the extrinsics of a multi-LiDAR system by combining motion information with surrounding appearance.

The pipeline of our proposed calibration system consists of three phases, as illustrated in Fig. 2. The first phase takes point clouds as input, and results in the incremental motions of each LiDAR within a time interval $[k-1, k]$ (Sect. IV-A). The second phase initializes $\mathbf{T}_{l^i}^{l^1}$ using a least-squares solution (Sect. IV-B). Finally, the third phase utilizes the appearance cues in surroundings to register different LiDARs for refinement (Sect. IV-C).

IV. METHODOLOGY

A. Motion Estimation

To calculate a set of incremental motion between consecutive frames of each LiDAR, the LeGO-LOAM algorithm [15] is used. This method makes use of line and edge features in environments to estimate sensors' ego-motion. In our implementation, the individual transformations of all the sensors and extracted ground points are used to provide constraints to the extrinsic parameters, which are discussed in Sect. IV-B and Sect. IV-C.

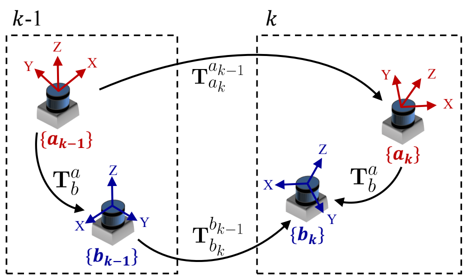


Fig. 3. The transformations between different LiDARs at $[k-1, k]$.

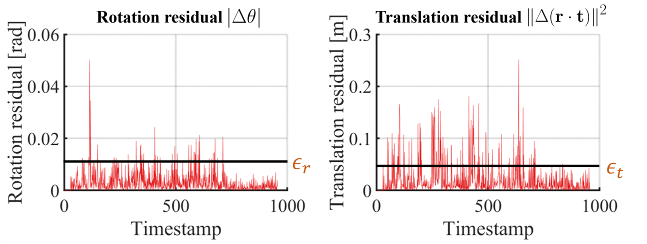


Fig. 4. An example of the screw motion residuals in rotation and translation of a set of estimated motion. ϵ_r, ϵ_r are the thresholds to filter the outliers.

B. Initialization

With l^1 as the reference sensor, we present a method of calibrating l^1 with l^i pairwise. To simplify the notations, we replace $\{l^1\}, \{l^i\}$ with $\{a\}, \{b\}$ to indicate the coordinate system of the reference sensor and the target sensor respectively. The constant transformation of two LiDARs can be initialized by aligning their estimated motion. Fig. 3 depicts the relationship between the motion of two LiDARs and their relative transformation. As the vehicle moves, the extrinsic parameters can be recovered using these poses for any k:

$$\mathbf{T}_{a_k}^{a_{k-1}}\mathbf{T}_b^a = \mathbf{T}_b^a\mathbf{T}_{b_k}^{b_{k-1}}, \tag{1}$$

where (1) can be decomposed in terms of its rotation and translation components with the following equations:

$$\mathbf{R}_{a_k}^{a_{k-1}}\mathbf{R}_b^a = \mathbf{R}_b^a\mathbf{R}_{b_k}^{b_{k-1}}, \tag{2}$$
$$(\mathbf{R}_{a_k}^{a_{k-1}} - \mathbf{I}_3)\mathbf{t}_b^a = \mathbf{R}_b^a\mathbf{t}_{b_k}^{b_{k-1}} - \mathbf{t}_{a_k}^{a_{k-1}}. \tag{3}$$

The method described in [1] is used to solve these two equations. Based on (2), the pitch-roll rotation can be calculated directly using the estimated rotations, while the yaw rotation, the translation can be computed using (3).

1) Outlier Filter: The poses of sensors have two constraints that are independent of the extrinsic parameters, which were presented as the screw motion in [16]:

$$\theta_{a_k}^{a_{k-1}} = \theta_{b_k}^{b_{k-1}} \tag{4}$$
$$\mathbf{r}_{a_k}^{a_{k-1}}\cdot\mathbf{t}_{a_k}^{a_{k-1}} = \mathbf{r}_{b_k}^{b_{k-1}}\cdot\mathbf{t}_{b_k}^{b_{k-1}}, \tag{5}$$

where θ denotes the angle of a rotation matrix $\mathbf{R}$, and $\mathbf{r}$ is the corresponding rotation axis[2]. The screw motion residuals include rotation and translation residuals, which are calculated as: $|\theta_a - \theta_b|$, $\|\mathbf{r}_a\cdot\mathbf{t}_a - \mathbf{r}_b\cdot\mathbf{t}_b\|^2$.

We adopt the screw motion residuals to evaluate the performance of the previous motion estimation phase. Fig. 4 shows the screw motion residuals in a real-world example, where we find the estimated motion is very noisy. Hence, the outliers are filtered if both their rotation and translation residuals are larger than the thresholds: ϵ_r, ϵ_t.

2) Pitch-roll rotation computation: It is efficient to use rotation matrix to solve (2) since the orthogonal constraint should be considered. For this reason, we employ the quaternion ($\mathbf{q} = [q_w, q_x, q_y, q_z]^\top$) following Hamilton notation to represent rotation. (2) can be thus rewritten by substituting $\mathbf{q}_b^a = (\mathbf{q}_b^a)_z(\mathbf{q}_b^a)_{yx}$ as below:

$$\begin{aligned} &\mathbf{q}_{a_n}^{a_{n-1}} \otimes (\mathbf{q}_b^a)_{yx} = (\mathbf{q}_b^a)_{yx} \otimes \mathbf{q}_{b_n}^{b_{n-1}} \\ \Rightarrow &\left[\mathbf{Q}_1(\mathbf{q}_{a_n}^{a_{n-1}}) - \mathbf{Q}_2(\mathbf{q}_{b_n}^{b_{n-1}})\right]\cdot(\mathbf{q}_b^a)_{yx} \\ \Rightarrow &\mathbf{Q}_n^{n-1}\cdot(\mathbf{q}_b^a)_{yx} = \mathbf{0}, \end{aligned} \tag{6}$$

where

$$\begin{aligned} \mathbf{Q}_1(\mathbf{q}) &= \begin{bmatrix} q_w\mathbf{I}_3 + [\mathbf{q}_{xyz}]_\times & \mathbf{q}_{xyz} \\ -\mathbf{q}_{xyz}^\top & q_w \end{bmatrix} \\ \mathbf{Q}_2(\mathbf{q}) &= \begin{bmatrix} q_w\mathbf{I}_3 - [\mathbf{q}_{xyz}]_\times & \mathbf{q}_{xyz} \\ -\mathbf{q}_{xyz}^\top & q_w \end{bmatrix} \end{aligned} \tag{7}$$

are matrix representations for left and right quaternion multiplication, $[\mathbf{q}_{xyz}]_\times$ is the skew-symmetric matrix of $\mathbf{q}_{xyz} = [q_x, q_y, q_z]^\top$, and $\otimes$ is the quaternion multiplication operator.

With N pairs of filtered rotations, we are able to formulate an over-constrained linear system as follows:

$$\begin{bmatrix} \mathbf{Q}_1^0 \\ \vdots \\ \mathbf{Q}_N^{N-1} \end{bmatrix}_{4N\times 4}\cdot(\mathbf{q}_b^a)_{yx} = \mathbf{Q}_N\cdot(\mathbf{q}_b^a)_{yx} = \mathbf{0}, \tag{8}$$

Using SVD to decompose $\mathbf{Q}_N = \mathbf{USV}^\top$, $(\mathbf{q}_b^a)_{yx}$ is computed by the weighted sum of $\mathbf{v}_3$ and $\mathbf{v}_4$:

$$(\mathbf{q}_b^a)_{yx} = \lambda_1\mathbf{v}_3 + \lambda_2\mathbf{v}_4, \tag{9}$$

where $\mathbf{v}_3$ and $\mathbf{v}_4$ are the last two column vectors of $\mathbf{V}$, and λ_1, λ_2 are two scalars. Therefore, $(\mathbf{q}_b^a)_{yx}$ can be obtained by solving the following equations:

$$\begin{aligned} x_{(\mathbf{q}_b^a)_{yx}}y_{(\mathbf{q}_b^a)_{yx}} &= -z_{(\mathbf{q}_b^a)_{yx}}w_{(\mathbf{q}_b^a)_{yx}} \\ \|(\mathbf{q}_b^a)_{yx}\| &= 1, \end{aligned} \tag{10}$$

where $x_\mathbf{q}, y_\mathbf{q}, z_\mathbf{q}, w_\mathbf{q}$ are the elements of a quaternion.

3) Yaw rotation and translation computation: Due to our planar motion assumption, the translational offset on $z-$ axis is unobservable. Consequently, we set $t_z = 0$ and rewrite (3) by removing the third row as follows:

$$\mathbf{R}_1\begin{bmatrix} t_x \\ t_y \end{bmatrix} - \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) \\ \sin(\gamma) & \cos(\gamma) \end{bmatrix}\mathbf{t}_1 = -\mathbf{t}_2, \tag{11}$$

where t_x and t_y are unknown translations along $x-$ and $y-$ axes, and γ is the unknown rotation angle around $z-$ axis. $\mathbf{R}_1$ is the 2×2 upper-left submatrix of $(\mathbf{R}_{a_n}^{a_{n-1}} - \mathbf{I}_3)$, $\mathbf{t}_1 = [t_{11}, t_{12}]^\top$ are the first two elements of $\mathbf{R}_b^a\mathbf{t}_{b_n}^{b_{n-1}}$, and $\mathbf{t}_2$

[2]The rotation angle and axis are calculated using the $\log(\cdot)^\vee$ operator such that $\boldsymbol{\phi} = \log(\mathbf{R})^\vee, \boldsymbol{\phi} = \theta\mathbf{r}$.

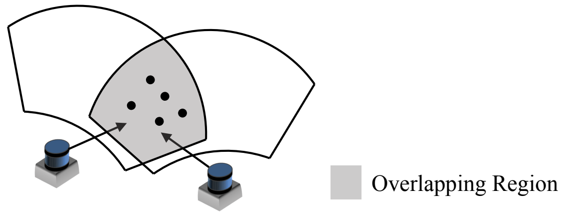


Fig. 5. The FOV of each LiDAR and overlapping region are visualized.

denote the first two elements of $\mathbf{t}_{a_n}^{a_{n-1}}$. We can rewrite (12) as a matrix-vector equation:

$$\underbrace{\begin{bmatrix}\mathbf{R}_1 & \mathbf{J}\end{bmatrix}}_{\mathbf{G}_{2\times 4}}\begin{bmatrix} t_x \\ t_y \\ -\cos(\gamma) \\ -\sin(\gamma)\end{bmatrix}, \quad \text{where } \mathbf{J}=\begin{bmatrix} t_{11} & -t_{12} \\ t_{12} & t_{11}\end{bmatrix}. \tag{12}$$

We can also construct a linear system from (12) with the filtered motion:

$$\underbrace{\begin{bmatrix}\mathbf{G}_1^0 \\ \vdots \\ \mathbf{G}_N^{N-1}\end{bmatrix}}_{\mathbf{A}_{2N\times 4}}\underbrace{\begin{bmatrix} t_x \\ t_y \\ -\cos(\gamma) \\ -\sin(\gamma)\end{bmatrix}}_{\mathbf{x}_{4\times 1}} = -\underbrace{\begin{bmatrix}(\mathbf{t}_2)_1^0 \\ \vdots \\ (\mathbf{t}_2)_N^{N-1}\end{bmatrix}}_{\mathbf{b}_{2N\times 1}}, \tag{13}$$

where $\mathbf{x}$ is obtained by applying the least-squares approach.

C. Refinement

In this section, we combine the coarse initialization results with the sensor measurements to refine the extrinsic parameters. Firstly, to recover the unknown t_z, the ground points are utilized. And then we estimate a set of transformations $\mathcal{T}_a^b = \{(\mathbf{T}_b^a)_k\}$ from $\{a\}$ to $\{b\}$ at each timestamp by registering $\mathcal{P}^{a_k}$ and $\mathcal{P}^{b_k}$. To improve the registration accuracy, we also apply an overlap filter to retain the points that lie in the overlapping regions of two sensors.

1) Ground planes alignment: In the motion estimation phase, we have extracted K pairs of ground points $\mathcal{G}^{a_k}, \mathcal{G}^{b_k}$ of the reference and target LiDARs. Since the segmented point clouds are noisy, we implement the random sample consensus (RANSAC) plane fitting algorithm to reject outliers. Denoting $\mathbf{c}_k^a, \mathbf{c}_k^b$ as the centroids of $\mathcal{G}^{a_k}, \mathcal{G}^{b_k}$ after filtering, we use the mean value of $\left(\mathbf{c}_k^a - \mathbf{R}_b^a\mathbf{c}_k^b\right)_z$ at each timestamp to determine t_z.

2) Overlap Filter: Precise registration between LiDARs is challenging since their overlapping field of view (FOV) is both limited and unknown. To tackle this issue, we employ an overlap filter to retains points that lie within the counterpart LiDARs' FOV, as the gray area depicted in Fig. 5. We denote the original point cloud captured by a and b at k as $\mathcal{P}^{a_k}, \mathcal{P}^{b_k}$ respectively. a point $\mathbf{p} \in \mathcal{P}^{b_k}$ can be transformed from $\{b\}$ to $\{a\}$ using the initial $\mathbf{T}_b^a$ as follows:

$$\widetilde{\mathbf{p}} = \mathbf{T}_b^a\mathbf{p}. \tag{14}$$

And then we employ the KD-Tree searching method

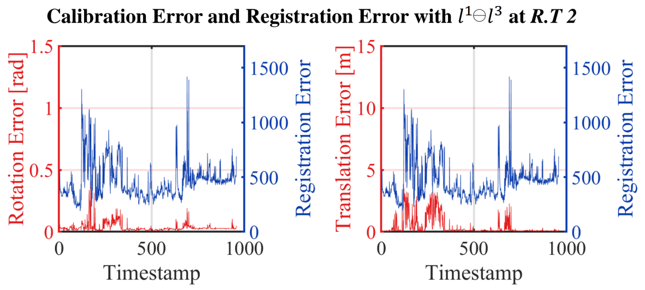


Fig. 6. This figure depicts the curves calibration error and registration error with $l^1 \ominus l^3$ at *R.T 2*.

realize the overlap filter to construct $\mathcal{S}^{a_k}, \mathcal{S}^{b_k}$:

$$\begin{aligned}\mathcal{S}^{a_k} &= \left\{\forall \mathbf{p}_1 \in \mathcal{P}^{a_k} : d(\mathbf{p}_1, \widetilde{\mathbf{p}}_2) < r, \exists \mathbf{p}_2 \in \mathcal{P}^{b_k}\right\} \\ \mathcal{S}^{b_k} &= \left\{\forall \mathbf{p}_2 \in \mathcal{P}^{b_k} : d(\widetilde{\mathbf{p}}_2, \mathbf{p}_1) < r, \exists \mathbf{p}_1 \in \mathcal{P}^{a_k}\right\},\end{aligned} \tag{15}$$

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are points, $d(\cdot,\cdot)$ is the Euclidean distance between two points, and r is a threshold. We define the overlap factor as:

$$\Omega_k = \frac{|\mathcal{S}^{a_k}|}{|\mathcal{P}^{a_k}|}\cdot\frac{|\mathcal{S}^{b_k}|}{|\mathcal{P}^{b_k}|}, \tag{16}$$

where $|\cdot|$ is the size of a set. The point clouds with $\Omega_k > 0.8$ are selected as the inputs for the following registration step.

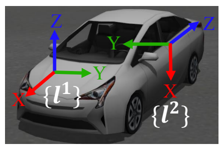
3) Registration: To estimate the relative transformations, the point-to-plane Iterative Closest Point (ICP) is used. After registering all the point cloud pairs captured at a different timestamp, we obtain a set of transformations $\mathcal{T}_b^a$. By computing the registration errors and calibration errors computed with the ground truth of each element, we find that their values are linear. An example of these errors over timestamp is depicted in Fig. 6, where the curves of registration error and calibration error have similar trends, especially the positions of the peak. Hence, we apply it to all calibration cases. In $\mathcal{T}_b^a$, only a series of transformations with the minimum registration error are selected as candidates, and the refinement results are computed as their mean values.

V. Experiment

In this section, we divide the evaluation into two separate steps. Firstly, the initial calibration experiments are presented with simulated data and real sensor sets. Then we test the refinement on real sensor data and demonstrate that the initial results can be improved using appearance cues.

A. Implementation Details

We use the ICP library [17] to process point clouds. With empirically setting parameters: $\epsilon_r = 0.01, \epsilon_t = 0.01, r = 10$, our method can obtain promising results. Since the success of motion-based calibration highly depends on the quality of motion which a vehicle undergoes, we design several paths with different rotations and scales to test our proposed algorithm. These paths include three simulated trajectories (*S.T 1-S.T 3*) on simulation and two real trajectories (*R.T 1-R.T 2*) on real sensor data. In the experiments, two platforms with different sensor setups are used.



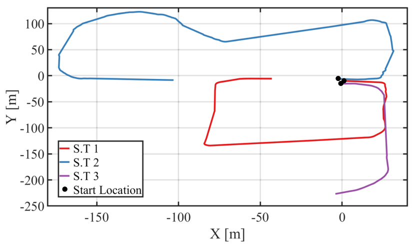


Fig. 7. (Left) The simulated platform and (right) the three trajectories which the platform follows.The rotation offset is about $[0, 3.14, 1.57]$ rads in roll, pitch, and yaw respectively. The corresponding translation are about $[-2.5, 1.5, 0]$ meters along $x-$, $y-$, and $z-$ axes respectively.

TABLE I
THE INITIAL CALIBRATION RESULTS WITH SIMULATED DATA.

σ^2	Trajectory	Rotation Error [rad]		Translation Error [m]	
		Kabsch	Proposed	Kabsch	Proposed
σ_1^2	*S.T 1*	0.10	**0.01**	**0.22**	0.28
	S.T 2	0.80	**0.00**	0.66	**0.28**
	S.T 3	0.08	**0.01**	0.80	**0.48**
σ_2^2	*S.T 1*	1.37	**0.07**	2.13	**1.25**
	S.T 2	0.94	**0.04**	1.80	**1.20**
	S.T 3	1.17	**0.02**	1.66	**1.44**

1) Simulation: The simulation software[3] is publicly available. For testing, we manually mount two sensors at different positions on the vehicle platform, as shown in Fig. 7 (left). The rotation offset between them is approximately $[0, 3.14, 1.57]$ rad in roll, pitch, and yaw, respectively. The corresponding translation are approximately $[-2.5, 1.5, 0]$ meters along $x-, y-$, and $z-$ axes respectively. We can thus acquire the positions of these sensors in the form of ground truth at 5 Hz. The refinement is not tested in simulation because this platform does not provide stable point clouds without time distortion.

2) Real Sensor: While our approach is not limited to a particular number of sensors, we are interested in calibrating between the reference LiDAR and two target LiDARs based on our platform. As shown in Fig. 1, three 16-beam RS-LiDARs[4] are rigidly mounted on the vehicle. The setup of this multi-LiDAR system has significant transformation among sensors. Especially, l^3 is mounted with approximately 180 degrees rotation offset in yaw. In later sections, we use $l^1 \ominus l^i$ to represent the configuration between l^1 and l^i. Since we do not know the precise extrinsic parameters, we use the parameters (shown in Table II) provided by the manufacturer to evaluate our proposed algorithm.

In evaluation, the error in rotation is measured by the angle difference between the ground truth and the resulting rotation, which is calculated as $e_r = \| \log(\mathbf{R}_{\text{gt}}\mathbf{R}_{\text{resulting}}^{-1})^{\vee}\|_2$. Similarly, the error in translation is computed using vector subtraction as $e_t = \|\mathbf{t}_{\text{gt}} - \mathbf{t}_{\text{resulting}}\|_2$. The translation error on $z-$ axis will not be counted of the initialization results because of the planar movement assumption.

B. Performance of Initialization

We take the modified Kabsch algorithm [2] that operates at matrix representation for comparison.

[3]https://github.com/osrf/car_demo/
[4]https://www.robosense.ai/rslidar/rs-lidar-16

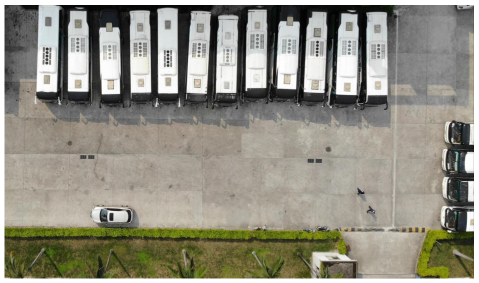


Fig. 8. The testing environment for our calibration experiments.

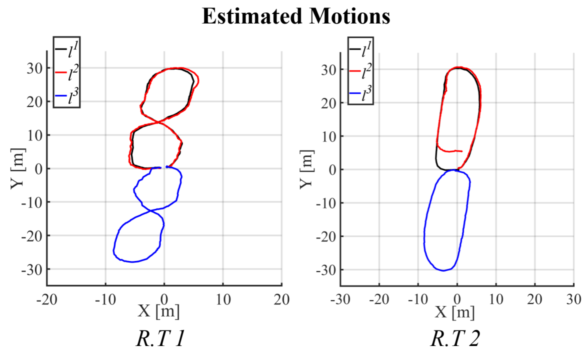


Fig. 9. The black, red, and blue lines indicate the estimated motionof l_1, l_2, l_3 respectively at *R.T 1-R.T 2*.

1) Simulation: The simulated trajectories *(S.T 1-S.T 3)* are visualized in Fig. 7 (right), where the third one is considered as the most challenging one since it has few rotations. To test the robustness of our proposed algorithm, the sensor's motion are added with zero-mean Gaussian noise $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2)$. σ^2 is set to two values: $\sigma_1^2 = 0.0001$ and $\sigma_2^2 = 0.001$ for evaluation. For each value, all simulated motion is tested. The calibration results are shown in Table I. The proposed algorithm can successfully initialize the rotation offset with low error and outperform the Kabsch algorithm in most of the cases. The translation error with σ_2^2 is larger than 1m, meaning that our initialization approach is not robust in noisy data.

2) Real Sensor: We carry out a real sensor experiment to validate the proposed method. Two trajectories (*R.T 1-R.T 2*) are designed in an urban environment (shown in 8), and we drive the vehicle to follow them. After estimating the individual motion of each LiDAR, we use the results to initialize the calibration. The estimated motion is depicted in Fig. 9, where we can observe that the calculated trajectory of l^2 at *R.T 2* drifts. The initialization results are presented in Table III. Our method performs well in recovering the rotation offset (< 0.15rad), but fail in calculating the translation offset (> 0.5m) in all cases. With the above results in simulation and real-world environments, we can conclude that the initialization phase can provide coarse estimates to the extrinsic parameters. For precise results, an additional refinement step is required.

C. Performance of Refinement

In our experiments, we select 10 transformations from $\mathcal{T}_b^a$ as the candidates for the optimal refinement results. We

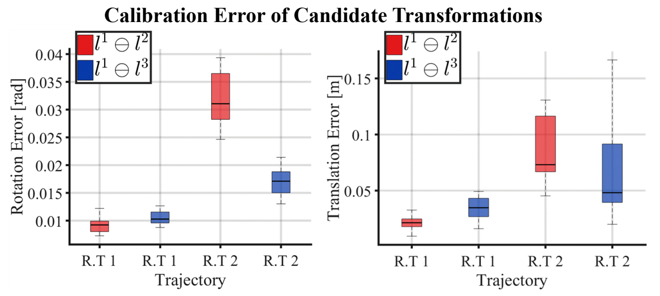


Fig. 10. The calibration errors in each case of the candidate. The means of them are small at *R.T 1*, but large at *R.T 2*.

TABLE II

THE CALIBRATION GROUND TRUTH OF OUR MULTI-LIDAR SYSTEM.

Conf.	Rotation [rad]			Translation [m]		
	x	y	z	x	y	z
$l^1 \ominus l^2$	0.01	0.08	0.03	0.42	0.00	-1.26
$l^1 \ominus l^3$	-0.02	0.01	-3.11	-2.11	0.06	-1.18

TABLE III

THE INITIALIZATION RESULTS.

Conf.	Traj.	Rotation Error [rad]		Translation Error [m]	
		Kabsch	Proposed	Kabsch	Proposed
$l^1 \ominus l^2$	*R.T 1*	0.94	**0.06**	1.60	**0.47**
	R.T 2	0.73	**0.03**	4.42	**1.95**
$l^1 \ominus l^3$	*R.T 1*	1.29	**0.14**	**1.23**	1.26
	R.T 2	0.60	**0.08**	**1.36**	2.04

TABLE IV

THE REFINEMENT RESULTS.

Conf. / Traj.		$l^1 \ominus l^2$ *R.T 1*	$l^1 \ominus l^2$ *R.T 2*	$l^1 \ominus l^3$ *R.T 1*	$l^1 \ominus l^3$ *R.T 2*
Rotation [rad]	x	0.01	0.01	-0.02	-0.02
	y	0.08	0.07	0.02	0.00
	z	0.04	0.04	-3.14	-3.13
	error	**0.01**	0.03	**0.01**	0.02
Translation [m]	x	0.43	0.46	-2.13	-2.07
	y	0.00	-0.01	0.09	0.08
	z	-1.26	-1.27	-1.18	-1.20
	error	**0.01**	0.08	0.03	0.04

plot their calibration errors in each case in Fig. 10. The detailed calibration results are shown in Table IV, where all the rotation and translation errors are less than 0.04rad and 0.1m, respectively. Compared with the result in Table III, the refinement phase can improve the estimated parameters.

D. Discussion

Since the proposed calibration method achieve accurate results, it do not perform well in the below cases. Firstly, the initialization highly relies on the accuracy of estimated motion, especially when the translational offset is imprecise. Secondly, We assume that LiDARs' views should have overlapping regions, Finally, the registration between point clouds may fail in several feature-less environments.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented a novel system for automatically calibrating of a multi-LiDAR system without any extra sensors, calibration target, or prior knowledge about surroundings. Our approach makes use of the complementary strengths of motion-based and appearance-based calibration methods. The individual motion of each LiDAR is estimated by an odometry algorithm. These poses are then utilized to initialize the extrinsic parameters. Finally, the results are refined by exploiting appearance cues in sensors' overlap. The performance of our method is demonstrated through a series of simulated and real-world experiments with reliable and accurate calibration results. There are several possible extensions to this work: (1) online, active multi-LiDAR calibration, (2) releasing the overlapping requirement, and (2) applications including localization and 3D object detection based on a multi-LiDAR system.

REFERENCES

[1] L. Heng, B. Li, and M. Pollefeys, "Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1793–1800.

[2] Z. Taylor and J. Nieto, "Motion-based calibration of multimodal sensor extrinsics and timing offset estimation," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1215–1229, 2016.

[3] L. Zhou, Z. Li, and M. Kaess, "Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences," in *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on*. IEEE, 2018.

[4] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments," in *Robotics: Science and Systems (RSS)*, 2018.

[5] P. Yun, L. Tai, Y. Wang, C. Liu, and M. Liu, "Focal loss in 3d object detection," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1263–1270, 2019.

[6] J. Jiao, Q. Liao, Y. Zhu, T. Liu, Y. Yu, R. Fan, L. Wang, and M. Liu, "A novel dual-lidar calibration algorithm using planar surfaces," *arXiv preprint arXiv:1904.12116*, 2019.

[7] M. He, H. Zhao, F. Davoine, J. Cui, and H. Zha, "Pairwise lidar calibration using multi-type 3d geometric features in natural scene," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1828–1835.

[8] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers." in *Robotics: Science and Systems*, vol. 2, 2013.

[9] Q. Liao, Z. Chen, Y. Liu, Z. Wang, and M. Liu, "Extrinsic calibration of lidar and camera with polygon," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019.

[10] M. He, H. Zhao, J. Cui, and H. Zha, "Calibration method for multiple 2d lidars system," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3034–3041.

[11] R. Horaud and F. Dornaika, "Hand-eye calibration," *The international journal of robotics research*, vol. 14, no. 3, pp. 195–210, 1995.

[12] K. Daniilidis, "Hand-eye calibration using dual quaternions," *The International Journal of Robotics Research*, vol. 18, no. 3, pp. 286–298, 1999.

[13] T. Qin and S. Shen, "Online temporal calibration for monocular visual-inertial systems," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3662–3669.

[14] Z. Yang and S. Shen, "Monocular visual-inertial fusion with online initialization and camera-imu calibration," in *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2015, pp. 1–8.

[15] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," 2018.

[16] H. H. Chen, "A screw motion approach to uniqueness analysis of head-eye geometry," in *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1991, pp. 145–151.

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