Single Frame Lidar-Camera Calibration Using Registration of 3D Planes.

Ashutosh Singandhupe, Hung Manh La *IEEE Senior Member*, Quang Ha *IEEE Senior Member*

***Abstract*— This work focuses on finding the extrinsic parameters (rotation and translation) between the Lidar and an RGB camera sensor. We use a planar checkerboard and place it inside the Field-of-View (FOV) of both the sensors, where we extract the 3D plane information of the checkerboard acquired from the sensor's data. The plane coefficients extracted from the sensor's data are used to construct a well-structured set of 3D points. These 3D points are then 'aligned,' which gives the relative transformation between the two sensors. We use our proposed method Correntropy Similarity Matrix Iterative Closest Point (CoSM-ICP) Algorithm to estimate the relative transformation. This work uses a single frame of the point cloud data acquired from the Lidar sensor and a single frame from a calibrated camera data to perform this operation. From the camera image, we use projection of the calibration target's corner points to compute the 3D points, and along the process, we calculate the 3D plane equation using the corner points. We evaluate our approach on a simulated dataset with complex environment settings since it has the freedom to assess under multiple configurations. Through results, we verify our method under various configurations.**

I. Introduction

Multi-sensor autonomous systems are generally designed on a predefined setup where the sensors are placed at known locations relative to each other. Since the sensors' relative translation and rotation components are already known, it does not necessarily involve any calibration methodology. However, modern autonomous systems with unique designs and sensors placed at different configurations demand automatic and efficient calibration methods for multi-sensor setups. Calibration in a multi-sensor system is essentially finding the relative transformation between the sensors. With our focus on autonomous navigation, lidar and camera configurations are explicitly designed to the requirement of the system and need to be calibrated (finding the relative rotation and translation between the two sensors) for efficient autonomous navigation [1]–[6].

Lidar and camera calibration is a well-studied problem, and most of the previous approaches require calibration cards or some well-defined calibration objects [1], [7]–[16]. Fig. 1 shows a simple lidar camera configuration setup. A calibration card is placed in the Field-of-View (FOV) of both the sensors. Extracting features from a checkerboard is convenient, making a planar checkerboard one of the most widely used calibration objects. Most of these approaches have emphasized placing the calibration objects in the FOV of both the sensors. Finding the correspondences between the lidar and the camera data points is crucial for efficient lidar and camera calibration [17]. The correspondences are calculated either manually or automatically using feature extraction algorithms. The accurate estimation of the correspondences is essential for efficient calibration.

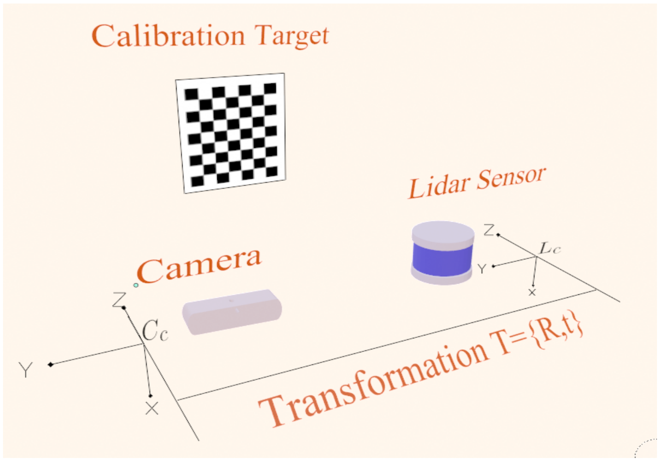


Fig. 1: Sample lidar and camera configuration setup (It is a stereo camera setup, however we use only the images from the left camera for our work). C_c is the camera coordinate frame (here we consider the left camera center of the stereo camera's coordinate frame), and L_c is the lidar coordinate frame. The objective here is to compute the transformation T between L_c and C_c.

The earlier method proposed by Zhang and Pless [7] aimed to calibrate a 2D laser and a monocular camera, and they used a planar checkerboard pattern to achieve that. Their approach involved placing the checkerboard in front of the 2D laser and the camera with different poses. The relative transformation is calculated by minimizing the re-projection error of features of the checkerboard as viewed by the 2D lidar and the camera. Their algorithm required at least five poses, whereas, in theory, the extrinsic calibration could be achieved in 3 poses. Unnikrishnan and Hebert [18] extended this method to a 3D laser, in which they estimate the plane parameters from the 3D lidar and the camera data. In their approach, the initial rotation and translation

This work is supported by the U.S. National Science Foundation (NSF) under grants NSF-CAREER: 1846513 and NSF-PFI-TT: 1919127, and the U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology (USDOT/OST-R) under Grant No. 69A3551747126 through INSPIRE University Transportation Center. The views, opinions, findings and conclusions reflected in this publication are solely those of the authors and do not represent the official policy or position of the NSF and USDOT/OST-R.

Ashutosh Singandhupe and Dr. Hung La are with the Advanced Robotics and Automation (ARA) Lab, Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA. Dr. Quang Ha is with the School of Electrical and Data Engineering, University of Technology Sydney, Australia.

The source code of our implementation is available at the ARA lab's github: `https://github.com/aralab-unr/Lidar-Cam-Calib` were calculated using plane-plane correspondence and later refined by minimizing the point to plane distance. Another work proposed by Geiger et al. [19], placed several checkerboards on the scene where their proposed algorithm detects the checkerboards and matches them with the planes from the laser data. Since there were several checkerboards in the scene, one shot of the scene is enough. Most of these methods involve the use of plane information of the checkerboard and do not consider the boundary of the checkerboard. Work by [20] requires fewer poses of the calibration target and has shown accurate results. In their work, the checkerboard is placed closer to the sensor, allowing the laser point to approximate the boundary more accurately, and hence it showed high accuracy. The authors introduced a similarity transformation between the lidar and the camera, which showed better results than rigid transformation.

Apart from the planar checkerboard, researchers also used several other calibration objects for the calibration method [21]–[24]. Work proposed in [10] used a polygonal board, and the calibration parameters were estimated using the vertices of the board. However, in the camera frame, it involved manual labeling of the correspondences. These correspondences were used to estimate the calibration parameters using a genetic algorithm-based approach. Other approaches [11]–[13] have used planar boards with rectangular holes, planar boards with circular holes, and other 3D objects. One of the drawbacks of using all the above methods is that it is laborious and time-consuming.

In addition to the approaches mentioned above, several target-less or no object-based calibration approaches have been proposed as well [25]–[27]. Initial work by Scaramuzza [28] has introduced a technique of using the natural scenes of calibrating a 3D laser and an omnidirectional camera. The features were extracted automatically from the sensor data based on their algorithm, and they manually selected the correspondence between the features. Another work by Moghadam [29] exploits the linear features present in the indoor environment. It uses the 3D line features from the point cloud and the corresponding 2D line segment from the camera to estimate the rigid body transformation between the two. Boughorbal [30] proposed to maximize the correlation between the sensor data using a chi-square test. This technique exploits the statistical dependence of the data acquired from the two sensors. Levinson and Thrun [31] uses a series of corresponding laser scans and camera images to estimate the calibration parameters. Work by [32] uses similar mutual information metrics between the two sensors' data by measuring the statistical dependence between the data [33].

Pandey, [32], proposed a theoretical derivation that estimates the kernel density of the probability distribution of the sensor data. Scott [33] proposed an approach for automatic calibration using diligently selected natural scenes. This algorithm searches over the chosen locations to extract models and yields better results. One advantage of this approach is that it requires no knowledge of the physical world and continuously finds scenes that constraints the optimization parameters. One recent and exciting approach by Jeong [34] exploits known features such as road markings. These features are captured by both the lidar and the stereo sensors and use a multiple cost function for robust optimization even with rough initial values. Work by Jeong [34] uses the road as a reference for computing transformation. However, it is required to be done in a controlled environment. It is also time-consuming and requires significant effort to obtain the transformation parameters. In this work, we propose an easy and efficient process to perform the lidar and camera calibration using a checkerboard calibration target. Our work is similar to the work presented by [35], where we compute the plane coefficients using the data from both the lidar and the camera data. Later on, these coefficients are used to construct a well-structured set of 3D points residing in that plane. We use our own proposed Correntropy Similarity Matrix (CoSM ICP) approach to align the constructed set of 3D points and compute the relative transformation between the sensors. In essence, the significant contributions of our work are outlined as follows:

- We develop our algorithm to find planes of the checkerboard pattern acquired from both the lidar and the camera sensor's data.
- We propose to compute the plane coefficients separately from both the sensor's data.
- From the plane coefficients acquired from both the sensors, we determine the plane's location and populate the plane with structured data points.
- Our proposed algorithm only needs to populate a limited number of points for the plane to plane matching.
- We leverage our CoSM ICP approach to perform the alignment and find the relative transformation.

The remaining paper is organized as follows: Section II describes our implementation of the proposed methodology. The results and evaluation of our proposed methodology are discussed in Section III. Discussion is given in Section IV. Conclusions are given in Section V.

II. Proposed Methodology

In this section, we describe the steps involved in our lidar-camera calibration. We perform different steps corresponding to the data acquired from the 3D lidar sensor and the camera data. The overall procedure involved in our work is shown in Fig. 2. Fig. 2, as read from left to right, initially shows the simulated scene setup that includes a lidar and a camera separated by a certain transformation. We acquire only a single frame of the lidar's 3D data and the camera's 2D image data, which contains the calibration target in its view. For the 3D lidar sensor, we manually select the region containing the 3D points corresponding to the calibration target (checkerboard). For the camera's 2D data, we assume that the camera is calibrated, which means

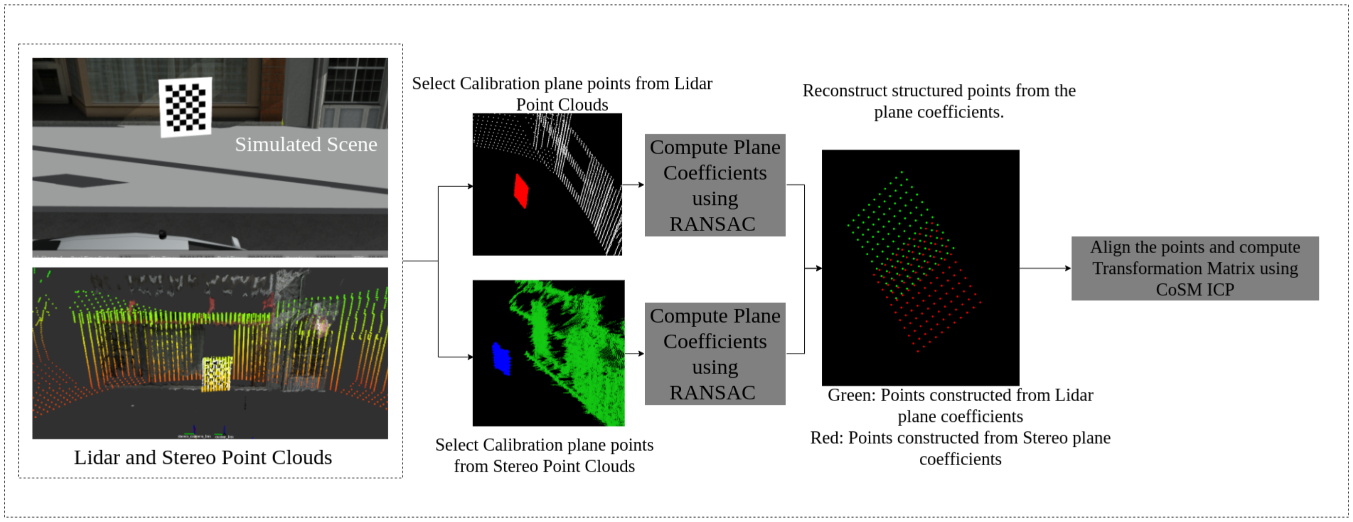

Simulated Scene
Lidar and Stereo Point Clouds
Select Calibration plane points from Lidar Point Clouds
Select Calibration plane points from Stereo Point Clouds
Compute Plane Coefficients using RANSAC
Compute Plane Coefficients using RANSAC
Reconstruct structured points from the plane coefficients.
Green: Points constructed from Lidar plane coefficients
Red: Points constructed from Stereo plane coefficients
Align the points and compute Transformation Matrix using CoSM ICP

Fig. 2: Flowchart of our approach.

that the camera's intrinsic parameters are already known. We project the 2D corner points to 3D points in the world frame. We discuss this process more in subsection II-B. It was convenient for us to use the same calibration target for performing the camera calibration as well. Again, the key point is to determine the plane coefficients acquired from the lidar's 3D data and the camera's projected 3D points. We compute the plane coefficients for the manually selected region of the lidar data using Random Sample Consensus (RANSAC). Using the camera's data, we compute the corner points of the checkerboard. We project the corners points in 3D space in the world frame using the intrinsic and extrinsic parameters, and we compute the plane coefficients of these 3D projected corner points. We construct a well-structured set of points using the plane coefficients, computed independently from the lidar and the camera data. We use our CoSM ICP approach to compute the transformation between these constructed point sets. The transformation computed by the CoSM ICP returns the relative transformation between the lidar and the camera sensors.

A. Lidar data processing

Using a single calibration target is relatively easy and intuitive for an extrinsic calibration between multiple sensors. As mentioned earlier, the first step for our process is to capture the 3D points of the lidar data. From the 3D data, we manually pick the region containing the points corresponding to the calibration target and ignore the rest. This selected data contains points of the calibration target, which is effectively a plane. Other automatic approaches, like distance filtering, can be used for this process, but we let the user pick the region for complete control. Now, from this 'selected' point set, we intend to get the corresponding plane coefficients. RANSAC is our choice for this process, which determines the best-fit plane of 3D points using inliers. We use the following steps for our work.

- Randomly selects 3 points from the 'selected' 3D lidar points.
- Compute the plane equation using these 3 points.
- Compute inliers using the computed plane with all other 3D points.
- Repeat the process with the highest inlier ratio.

For this setup, we set the maximum iterations for our RANSAC algorithm as 1000. 3D points that are within 10mm from the plane are considered inliers. The inlier ratio, which crosses 90% or more, is considered as a best-fit plane. The plane equation computed from the lidar points is given as $a_l x + b_l y + c_l z + d_l = 0$, where a_l,b_l,c_l and d_l represent the coefficients of the plane.

B. Camera data processing

This section details the process of computing the 3D plane equations of the calibration target using the 2D image data acquired from the camera. In this work, we assume that the camera is already calibrated. We know the intrinsic parameters $\alpha_x, \alpha_y, x_0, y_0$, where α_x,α_y represents the camera's focal length in terms of pixel dimensions and x_0, y_0 represent the principal point in the pixel dimensions. The parameters are described in the matrix form and are denoted as K. We ignore the skew, radial, and tangential distortion since we operate in a simulated environment. The chessboard corners are computed using OpenCV $findChessboardCorners$ function, which is based on the algorithm by Duda and Frese [36]. After the corner point detection, we use the well-known PnP algorithm to compute the pose of the calibration object from 3D-2D point correspondences. The result of PnP gives us the rotation and the translation components that transform a 3D point expressed in the object coordinate frame to the camera coordinate frame. For this method, we use an iterative method which is based on the Levenberg-Marquardt optimization method that minimizes the reprojection error. From the set of 3D computed corner points of the checkerboard pattern, we select three random points and transform them from the object frame to the camera frame using the rotation and the translation components computed from the PnP method. Let the 3 randomly selected be $p_1 =$

$\{x_1, y_1, z_1\}$,$p_2 = \{x_2, y_2, z_2\}$ and $p_3 = \{x_3, y_3, z_3\}$. We compute the normal vector $\vec{n}$ as,

$$\vec{n} = (p_2 - p_1) \times (p_3 - p_1) \tag{1}$$

where, $\times$ is the cross product in Equ.(1). The coefficients of the plane is then computed as,

$$\begin{aligned} a_c &= n.x, \\ b_c &= n.y, \\ c_c &= n.z, \\ d_c &= -(a * p_1.x + b * p_1.y + c * p_1.z). \end{aligned} \tag{2}$$

In Equ.(2), $*$ denotes multiplication and . represents the individual components in a vector. The plane equation computed from the camera points is given as $a_c x + b_c y + c_c z + d_c = 0$, where a_c, b_c, c_c and d_c represent the coefficients of the plane.

C. *Transformation estimation*

We compute the transformation between these sensors from the plane equations calculated from both the lidar and the camera data. For this step, we use the calculated plane equations to populate the planes with a fixed number of points separated by a known distance (e.g., 100 points). The point sets generated from this process can be 'aligned.' Fig.3 shows the structured set of points. The computed transformation gives us the transformation between the lidar and the camera sensor. The primary reason for this process is that the number of points in the lidar point set is different from the number of points in the computed 3D points from the camera. Our CoSM ICP needs to have an equal number of points for alignment. We describe our CoSM ICP approach to compute the transformation between the lidar and the camera data points.

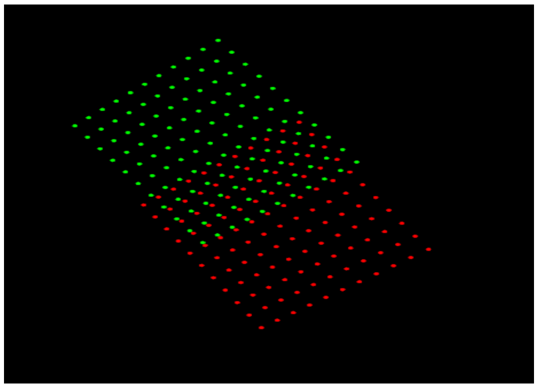


Fig. 3: Green is the *Source* point cloud $\mathbf{P_s}$ (as computed from lidar points). $\mathbf{P_s}$ contains N points $\mathbf{p}_j$ ($j = 1, .., N$), each of which is a 3D representation x_j, y_j, z_j. White is the *Target* point cloud $\mathbf{P_t}$ (as computed from camera data). $\mathbf{P_t}$ contains N points $\mathbf{q_k}$ ($k = 1, .., N$), each of which is a 3D representation x_k, y_k, z_k.

D. *Correntropy similarity matrix with iterative closest point algorithm*

We derive the idea of Correntropy from our previous work and extend this idea to the well known ICP algorithm [37]. Traditional ICP [38] describes the alignment of point clouds such that the Mean Square Error (MSE) between point sets is minimized. The MSE is the key criterion of convergence in ICP and its variants. ICP revolves around the problem of aligning points sets $\mathbf{P_s}$ and $\mathbf{P_t}$. We denote the point sets as $\mathbf{P}_s = \{[\mathbf{p}_j]_{j=1}^N\}$ (here $\{\}$ denotes a point set) and $\mathbf{P}_t = \{[\mathbf{q}_k]_{k=1}^N\}$, respectively. Our goal is to best align *Source* point set $\mathbf{P_s}$ to *Target* point set (or a model point set) $\mathbf{P_t}$. This leads to finding the appropriate rotation and translation between the two point sets, such that the Root Mean Square Error (RMSE) error between the two point set is minimized (or is within a defined threshold). This process is commonly known as registration between the *Source* $\mathbf{P_s}$ and the *Target* $\mathbf{P_t}$. Now, the problem can be written as: find the best $\mathbf{R}$ and $\mathbf{tr}$ such that the MSE, ε^2, between two point clouds ($\mathbf{P_s}$ and $\mathbf{P_t}$) is minimized. This problem can be mathematically written as,

$$\varepsilon^2 = \sum_{j,k=1}^{N} \left\| \mathbf{p}_j - (s\mathbf{R}\mathbf{q}_k + \mathbf{tr}) \right\|^2, \tag{3}$$

where s is the scaling component. $\mathbf{p}_j \in \mathbf{P_s}$ is the set of all the points in the *Source* point set, and $\mathbf{q}_k \in \mathbf{P}_t$ is the set of all the points in the *Target* point set ($j, k = 1, ..., N$). Our work is built on the work of Arun et al. [38], and it is well known from their work that initially they compute the corresponding points between *Source* and *Target* point sets. The Corresponding points in this work are computed as shortest point from each point in the *Source* dataset to the *Target* dataset. This can be efficiently done using $k-d$ tree data structure. We introduce an additional parameter using the Correntropy criterion to find the similarity metric between points. Now, we say that for a *Source* point index i the corresponding point index in the *Target* point set is $\mathbf{c}(i)$ ($\mathbf{c}$ is vector of corresponding points from *Source* to *Target*. Note: we do not perform reciprocal correspondence, i.e., from *Target* to *Source*). We compute the similarity between the two corresponding points $\mathbf{P_s}$ and $\mathbf{P_t}$ as

$$\begin{aligned} \mathbf{d} &= \mathbf{P_s}(i) - \mathbf{P_t}(\mathbf{c}(i)), \\ G_\sigma(\mathbf{P_s}(i) - \mathbf{P_t}(\mathbf{c}(i))) &= \frac{1}{(2\pi\sigma)^{\frac{1}{D}}} exp(\frac{\mathbf{d}^T\mathbf{d}}{2\sigma^2}). \end{aligned} \tag{4}$$

Here, D is the dimensions and $D = 3$ in this case [39]. Now, we can create a similarity matrix between the *Source* and the *Target* points based on the similarity metric computed in Equ. 4. Intuitively, the size of the similarity matrix is $N \times N$. In every iteration we initialize the value of this similarity matrix as zeros and update them in every iteration based on the similarity metric computed above in Equ.4. It is updated as shown below:

$$\begin{aligned} \mathbf{SM}(i, \mathbf{c}(i)) &= G_\sigma(\mathbf{P_s}(i) - \mathbf{P_t}(\mathbf{c}(i))), \\ \mathbf{SM}(\mathbf{c}(i), i) &= G_\sigma(\mathbf{P_s}(i) - \mathbf{P_t}(\mathbf{c}(i))), \end{aligned} \tag{5}$$

where $\mathbf{SM}$ is the similarity matrix which we intend to use in the computation of the rotation component. The centroid of both the point clouds, i.e., the *Source* and *Target* is computed as given by Equ.6

$$\mathbf{s_{cen}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i, \\ \mathbf{t_{cen}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{q}_i, \tag{6}$$

where $\mathbf{s_{cen}}$ and $\mathbf{t_{cen}}$ are the computed centroid of *Source* and *Target* data set, respectively. The difference between the computed centroid $\mathbf{s_{cen}}$ and the individual *Source* points is computed. The same procedure is followed for the *Target* points. It is given in Equ. 7

$$\mathbf{p}_i^{'} = \mathbf{p}_i - \mathbf{s_{cen}}, \\ \mathbf{q}_j^{'} = \mathbf{q}_j - \mathbf{t_{cen}}. \tag{7}$$

The Singular Value Decomposition (SVD) for finding the rotation component is based on first finding the $4x4$ matrix as shown in Equ. 8 where we introduce our Similarity matrix:

$$\mathbf{H} = \sum_{i=1}^{N}\mathbf{p}_i^{'}\mathbf{SM}\mathbf{q}_i^{'T}. \tag{8}$$

The size of the **SM** matrix is $N \times N$ where as the size of the **p**' matrix is $4 \times N$ (under homogeneous transformation) which results in $4 \times N$ matrix. The result when multiplied with $\mathbf{q}'^T$ (which is a $N \times 4$ matrix) will have the final result in a 4×4 matrix form which is the size of the matrix **H**. The rest of the algorithm is the same as the traditional algorithm where we compute the SVD of **H** as given by Equ. 9

$$\mathbf{H} = \mathbf{U}\Lambda\mathbf{V}^T. \tag{9}$$

Now the Rotation component is calculated as given by Equ. 10.

$$\mathbf{R} = \mathbf{V}\mathbf{U}^T. \tag{10}$$

The determinant of R must be a positive integer. The translation component is simply computed as the difference of the centroids computed in Equ.7. Algorithm 2 describes the entire procedure of our proposed method.

III. Results

We performed our initial evaluation in a simulated environment provided by Open Robotics [40]. To demonstrate our results, we start with a basic simulated dataset containing simulated lidar data and a simulated camera setup (mounted on a simulated Prius model car) established in a simulated environment in Gazebo, which is shown in Fig.4 (a) and (b) [41]. The primary reason for selecting this simulation setup is to compare our results since we know the ground truth. This setup contains other environmental complexities like buildings and cars (apart from the calibration card) and simulated lighting conditions. Our experiments test the results with a linear transformation ranging from $0.05m$ to $2.5m$ (along x, y, z axes). We set up a simple test case of lidar-camera setup where the stereo camera faces the calibration target and is placed near the lidar sensor under

Algorithm 1 CoSM Algorithm for Lidar-Stereo Camera Calibration

```
1: function ReadDataSets($\mathbf{P_s}$,$\mathbf{P_t}$)   ▷ Read the Source and the Target datasets.
2:     while not converged do
3:         $\mathbf{c} \leftarrow$ ComputeCorrespondence($\mathbf{P_s}, \mathbf{P_t}$).   ▷ Compute Correspondence
4:         SM $= zeros(N, N)$   ▷ Initialize $N \times N$ Similarity Matrix to all zeros.
5:         for $i \leftarrow 1$ to $N$ do
6:             $\mathbf{d} = \mathbf{P_s}(i) - \mathbf{P_t}(\mathbf{c}(i))$.
7:             Compute $G_\sigma$ as shown in Equ.4.
8:             SM$(i, \mathbf{c}(i)) = G_\sigma(\mathbf{P_s}(i) - \mathbf{P_t}(\mathbf{c}(i)))$.
9:             SM$(\mathbf{c}(i), i) = G_\sigma(\mathbf{P_s}(i) - \mathbf{P_t}(\mathbf{c}(i)))$.
10:        end for
11:        Compute centroid as given in Equ.6.
12:        Compute the difference as given in Equ.7.
13:        Compute the H Matrix as given in Equ.8.
14:        Compute SVD of H as given in Equ.9.
15:        Compute R as given in Equ.10.
16:        Compute tr as difference of centroids.
17:    end while
18: end function
```

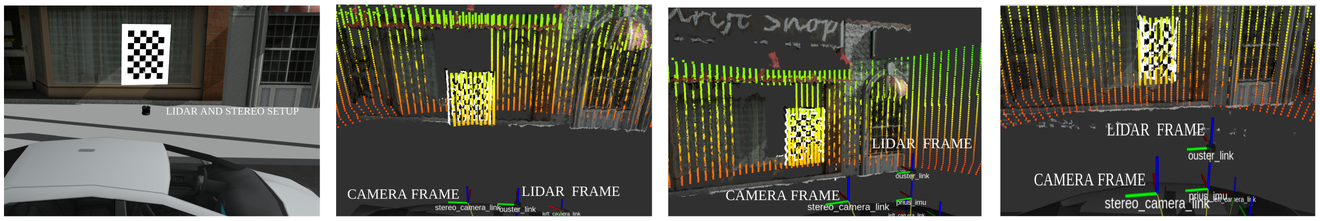
Algorithm 2 Algorithm to generate shot noise/impulse noise.

```
1: function GenerateOutliers($min, max$)▷ Function to Generate outliers.
2:     N=100
3:     meanidx=GenerateRandomNumers(min,max,N)
4:     varianceidx=GenerateRandomNumbers(0.01,100,N)
5:     for $i \leftarrow 1$ to $N$ do
6:         vals=GenerateNumbers(meanidx[i],variancedx[i])
7:         SaveFile(m1)
8:     end for
9:     noisevals=PickRandomVals(vals)
```

various configurations. (e.g., 5cm along the y axis of the lidar sensor or $\mathbf{t} = [0, 0.05, 0.0]$). Here, we intend to calculate the transformation between the camera (C_c) with respect to the lidar's frame L_c. The camera has a resolution of 1280×720, and since we use a simulated setup, we ignore the radial and tangential distortion (both were set to 0). In this scenario, we can evaluate our approach on multiple configurations where the ground truth is already known, as it can be seen in Fig. 4. So, in this case, our problem statement is defined as finding the transformation between the Lidar(L_c) and the camera(C_c) setup using our proposed methodology. This setup provides a base test case to verify our algorithm. Thus it can extend to complex real-time test cases.

A. Evaluation on Simulated data

The various configurations under which we performed our experiments are shown in Table I. It shows the original ground truth transformations between the lidar and the camera sensor. We place the calibration target at an average distance of 2m to 3m from the calibration target throughout



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

Fig. 4: Simulation setup for evaluating lidar-camera calibration under multiple configurations: (a) denotes a gazebo simulation of Prius car model with lidar and stereo camera; (b) denotes the TF-frames of the lidar and camera. The link $ouster\_link$ is the reference frame for lidar; (c) and (d) denote TF-frames of multiple configurations of lidar-camera setup. $\mathbf{t} = [t_x, t_y, t_z]$ denotes the translation component along x, y, z. It essentially denotes how the camera is transformed with respect to the lidar sensor along x, y, z. For (c) the translation between the lidar and the stereo sensor is $\mathbf{t} = [0, 0.5, 0.0]$, and for (d) it is $\mathbf{t} = [-0.5, 0.5, 0]$.

our experiments (this distance is with respect to the lidar coordinate frame (L_c)).

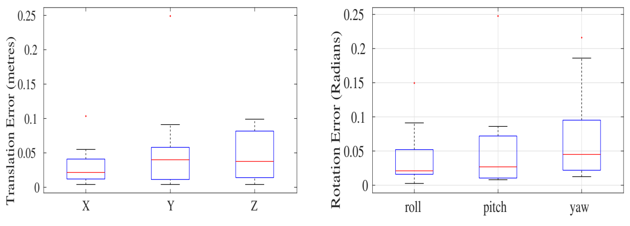
TABLE I: Configuration setups used in our experiments (or Ground Truth).

Setting	$t_x(m)$	$t_y(m)$	$t_z(m)$	roll (rad)	pitch(rad)	yaw(rad)
1	0	0.4	0.0	0.0	0.0	0.0
2	0	0.1	0.0	0.0	0.0	0.0
3	0	0.05	0.0	0.0	0.0	0.0
4	0	0.6	0.0	0.0	0.0	0.0
5	0	1.2	0.0	0.0	0.0	0.0
6	0	2.2	0.0	0.0	0.0	0.0
7	0.5	0.5	0.0	0.0	0.0	0.0
8	-0.5	1.5	0.3	0.0	0.0	0.0
9	-0.5	0.5	0	0.0	0.0	-0.0
10	0.5	0.5	0	0	0	-0.523599
11	0.5	0.5	0.3	0	0.349066	-0.523599
12	0.3	0.6	0.4	0	0.349066	-0.523599
13	0.2	0.3	0.2	0.261799	0	-0.523599
14	0.7	0.2	0.9	0	0.349066	0

For further evaluation, we show the average error in multiple configurations as done in the experiments. Fig.5 shows the average error of the individual components of the rotation and translation components under various configurations. From the experiments performed and from Fig. 5, we can see that under simple translation along x, y, z axes, our algorithm performs well (with individual RMSE's $\sim$ 0.01 along with all the individual components). Even under small rotations, our approach returns relatively close values (RMSE $\sim$ 0.02) compared to the ground truth. However, when the transformation between C_c and L_c is significant (>60 degrees or 1.0472 radians), the overall RMSE increases and the computed transformation quite far off from the ground truth values (RMSE >6.891).

One of the critical observations here was to estimate 3D points computed from the camera setup accurately. Throughout the experiments, we collected datasets for all the individual configurations as mentioned in Table I. We hand-picked each dataset that had a good collection of 3D point sets with less visible outliers. However, we hope that with further advancements in 3D depth estimation, we can get more accurate and robust results.

Since it was a simulated setup, it could have been simple



(a) (b)

Fig. 5: Translation and Rotation errors of individual components (x, y, z and roll,pitch and yaw) under various configurations compared to ground truth.

to have a lidar sensor, a camera setup, and a calibration target. However, one of our primary goals was to have a calibration technique under various environments. We plan to evaluate our approach in real-time; however, valuating a lidar and a camera setup under different configurations is time-consuming. So, in our simulated environment, we add environmental complexities and evaluate our approach. We let the user select the region in the 3D point sets corresponding to the calibration plane. Since it is essential for the algorithm to have accurate plane coefficients, manual selection can provide complete control.

IV. Conclusions

This work proposes a novel algorithm for an efficient Lidar-Camera calibration using a single frame from the lidar data and 3D points estimated from the camera data. Our approach can avoid using multiple poses of the calibration target since it only needs a single frame of the lidar and camera data. The later steps involve manually selecting 3D lidar points and estimating the plane coefficients from the lidar data, and automatically detecting the planes using the checkerboard corners from the camera data. From the computed plane coefficients (from both the sensor's data), we construct a well-spaced 3D point structure. Later, we propose to use our methodology called CoSM ICP to compute the transformation between the 'structured' points, thereby accomplishing the purpose of Lidar-Camera calibration. CoSM ICP is robust to large rotations and translations, which aided in computing the transformation between the lidar and the camera frame. The results show a positive approach towards a single frame lidar-camera calibration. Moreover, our proposed CoSM-ICP approach has shown promising results in the transformation computation between the lidar and the camera sensor. It would benefit the community to validate our approach for their own lidar-camera setup in real-time environments.

REFERENCES

[1] L. Huang and M. Barth, "A novel multi-planar lidar and computer vision calibration procedure using 2d patterns for automated navigation," in *2009 IEEE Intelligent Vehicles Symposium*. IEEE, 2009, pp. 117–122.

[2] A. Sehgal, A. Singandhupe, H. La, A. Tavakkoli, and S. Louis, "Lidar-monocular visual odometry with genetic algorithm for parameter optimization," in *The 14th International Symposium on Visual Computing (ISVC), Oct. 7-9, Lake Tahoe, NV, USA*, 10 2019, pp. 358–370.

[3] A. Singandhupe and H. La, "A review of slam techniques and security in autonomous driving," in *Proceedings of the 3rd IEEE International Conference on Robotic Computing (IRC), February 25-27, Naples, Italy*, 02 2019, pp. 602–607.

[4] J. Beltrán, C. Guindel, and F. García, "Automatic extrinsic calibration method for lidar and camera sensor setups," *CoRR*, vol. abs/2101.04431, 2021. [Online]. Available: https://arxiv.org/abs/2101.04431

[5] P. An, T. Ma, K. Yu, B. Fang, J. Zhang, W. Fu, and J. Ma, "Geometric calibration for lidar-camera system fusing 3d-2d and 3d-3d point correspondences," *Opt. Express*, vol. 28, no. 2, pp. 2122–2141, Jan 2020. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-28-2-2122

[6] J. Shi, Z. Zhu, J. Zhang, R. Liu, Z. Wang, S. Chen, and H. Liu, "Calibrcnn: Calibrating camera and lidar by recurrent convolutional neural network and geometric constraints," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10 197–10 202, 2020.

[7] Qilong Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2301–2306 vol.3.

[8] O. Naroditsky, A. Patterson, and K. Daniilidis, "Automatic alignment of a camera with a line scan lidar system," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3429–3434.

[9] S. Verma, J. S. Berrio, S. Worrall, and E. Nebot, "Automatic extrinsic calibration between a camera and a 3d lidar using 3d point and plane correspondences," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3906–3912.

[10] A. Dhall, K. Chelani, V. Radhakrishnan, and K. M. Krishna, "Lidar-camera calibration using 3d-3d point correspondences," *arXiv preprint arXiv:1705.09785*, 2017.

[11] C. Guindel, J. Beltrán, D. Martín, and F. García, "Automatic extrinsic calibration for lidar-stereo vehicle sensor setups," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6.

[12] Y. Park, S. Yun, C. Won, K. Cho, K. Um, and S. Sim, "Calibration between color camera and 3d lidar instruments with a polygonal planar board," *Sensors (Basel, Switzerland)*, vol. 14, pp. 5333–5353, 03 2014.

[13] S. Debattisti, L. Mazzei, and M. Panciroli, "Automated extrinsic laser and camera inter-calibration using triangular targets," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 696–701.

[14] X. Xu, L. Zhang, J. Yang, C. Liu, Y. Xiong, M. Luo, Z. Tan, and B. Liu, "Lidar–camera calibration method based on ranging statistical characteristics and improved ransac algorithm," *Robotics and Autonomous Systems*, vol. 141, p. 103776, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889021000610

[15] Y. Zhu, C. Li, and Y. Zhang, "Online camera-lidar calibration with sensor semantic information," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4970–4976.

[16] Y. Su, Y. Ding, J. Yang, and H. Kong, "A two-step approach to lidar-camera calibration," in *2020 25th International Conference on Pattern Recognition (ICPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jan 2021, pp. 6834–6841. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICPR48806.2021.9412085

[17] F. M. Mirzaei, D. G. Kottas, and S. Roumeliotis, "3d lidar–camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization," *The International Journal of Robotics Research*, vol. 31, pp. 452 – 467, 2012.

[18] R. Unnikrishnan and M. Hebert, "Fast extrinsic calibration of a laser rangefinder to a camera," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-09*, 01 2005.

[19] A.Geiger, F.Moosmann, O.Car, and B.Schuster, "Automatic camera and range sensor calibration using a single shot," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 3936–3943.

[20] L. Zhou, Z. Li, and M. Kaess, "Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5562–5569.

[21] Z. Pusztai and L. Hajder, "Accurate calibration of lidar-camera systems using ordinary boxes," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 394–402.

[22] T. Tóth, Z. Pusztai, and L. Hajder, "Automatic lidar-camera calibration of extrinsic parameters using a spherical target," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 8580–8586.

[23] Z. Bai, G. Jiang, and A. Xu, "Lidar-camera calibration using line correspondences," *Sensors*, vol. 20, p. 6319, 11 2020.

[24] M. Velas, M. Spanel, Z. Materna, and A. Herout, "Calibration of rgb camera with velodyne lidar," 2014.

[25] L. Heng, M. Bürki, G. Lee, P. Furgale, R. Siegwart, and M. Pollefeys, "Infrastructure-based calibration of a multi-camera rig," 05 2014, pp. 4912–4919.

[26] C. Häne, L. Heng, G. Lee, F. Fraundorfer, P. Furgale, T. Sattler, and M. Pollefeys, "3d visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection," *Image and Vision Computing*, 08 2017.

[27] K. Irie, M. Sugiyama, and M. Tomono, "Target-less camera-lidar extrinsic calibration using a bagged dependence estimator," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, 2016, pp. 1340–1347.

[28] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 4164–4169.

[29] P. Moghadam, M. Bosse, and R. Zlot, "Line-based extrinsic calibration of range and image sensors," vol. 2, 05 2013.

[30] F. Boughorbel, D. Page, C. Dumont, and M. Abidi, "Registration and integration of multisensor data for photorealistic scene reconstruction," *Proc SPIE*, 03 2001.

[31] J. Levinson and S. Thrun, *Unsupervised Calibration for Multi-beam Lasers*, 01 2014, pp. 179–193.

[32] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *Journal of Field Robotics*, vol. 32, 09 2014.

[33] T. Scott, A. A. Morye, P. Piniés, L. M. Paz, I. Posner, and P. Newman, "Choosing a time and place for calibration of lidar-camera systems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4349–4356.

[34] J. Jeong, Y. Cho, and A. Kim, "The road is enough! extrinsic calibration of non-overlapping stereo camera and lidar using road information," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2831–2838, 2019.

[35] E.-s. Kim and S.-Y. Park, "Extrinsic calibration between camera and lidar sensors by matching multiple 3d planes," *Sensors*, vol. 20, no. 1, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/1/52

[36] A. Duda and U. Frese, "Accurate detection and localization of checkerboard corners for calibration," 09 2018.

[37] A. Singandhupe, H. M. La, T. D. Ngo, and V. A. Ho, "Registration of 3d point sets using correntropy similarity matrix," *CoRR*, vol. abs/2107.09725, 2021. [Online]. Available: https://arxiv.org/abs/2107.09725

[38] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 5, p. 698–700, May 1987. [Online]. Available: https://doi.org/10.1109/TPAMI.1987.4767965

[39] J. Principe and J. Iii, "Information-theoretic learning," *Advances in unsupervised adaptive filtering*, 09 2000.

[40] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[41] O. S. R. Foundation, *Vehicle and city Simulation*, 2017-10-26. [Online]. Available: http://gazebosim.org/blog/car_sim