Multi-FEAT: Multi-Feature Edge Alignment for Targetless Camera-LiDAR Calibration

Bichi Zhang, Student Member, IEEE, Raj Thilak Rajan, Member, IEEE

Abstract—The accurate environment perception of automobiles and UAVs (Unmanned Ariel Vehicles) relies on the precision of onboard sensors, which require reliable in-field calibration. This paper introduces a novel approach for targetless camera-LiDAR extrinsic calibration called Multi-FEAT (Multi-Feature Edge AlignmentT). Multi-FEAT uses the cylindrical projection model to transform the 2D(Camera)-3D(LiDAR) calibration problem into a 2D-2D calibration problem, and exploits various LiDAR feature information to supplement the sparse LiDAR point cloud boundaries. In addition, a feature matching function with a precision factor is designed to improve the smoothness of the solution space. The performance of the proposed Multi-FEAT algorithm is evaluated using the KITTI dataset, and our approach shows more reliable results, as compared with several existing targetless calibration methods. We summarize our results and present potential directions for future work.

Index Terms—LiDAR, camera, sensor fusion, extrinsic calibration, autonomous systems

I. INTRODUCTION

Multi-sensor fusion techniques are applied to autonomous vehicles to improve the perception and localization tasks of any given agent. One of the typical combinations for environment perception is the fusion of camera and LiDAR, which is attractive since they have complementary characteristics. LiDARs can measure 3D positions of the surroundings accurately, while a camera can obtain rich color and texture information. Hence, a fusion of these sensors significantly improves the quality of the environment perception of any given autonomous system. A critical challenge for achieving multi-sensor fusion is to calibrate multiple sensors. For example, extrinsic calibration of a camera aims to find the spatial relationship between sensors characterized by rotations and translation parameters. Traditionally, extrinsic calibration methods use known reference objects in the field, e.g., checkerboards, to align common features in different sensor modalities and require multi-view frames [1]. In addition to checkerboard, some other shapes of objects are also developed to achieve camera-LiDAR calibration [2] [3]. Furthermore, other approaches use multiple boards to avoid working in multiple frames, and/or with people moving the target around to provide multiple frames [4] [5] [6]. Although target-based methods have provided reliable solutions in the past decades, human intervention is the major drawback for automotive systems, motivating the need for targetless calibration methods.

In this work, we leverage multiple features in the field of view (FOV) of LiDAR for better construction of the edge map, which is a 2D image carrying the edge information of the 3D point cloud. Our proposed pipeline that exploits the enriched multi-feature edge map is named Multi-FEAT: Multi-Feature Edge AlignmentT, which is briefly illustrated in Fig 1. The pipeline shows multiple features being extracted from the LiDAR point cloud, after a cylindrical projection, and aligned with the pre-processed camera output, to provide a joint calibration result using only one-shot (or snapshot, or single-shot) information. In summary, our key contributions in this work are as follows:

- We propose a targetless and unsupervised calibration method called Multi-FEAT, which solves the camera-LiDAR extrinsic calibration problem by exploiting multiple features, which are extracted from the point cloud.
- An occlusion-free cylindrical projection algorithm is developed for point cloud data, reducing the edge alignment error by eliminating the low-quality edges influenced by the occlusion effect.
- A cost function is designed to align the edge intensities of the camera images’ edge probability acquired from the multi-feature LiDAR point cloud, and propose a gradient ascent method to estimate the unknown extrinsic parameters.
- We compare our proposed Multi-FEAT algorithm with the state-of-the-art targetless calibration methods, using the open-source KITTI dataset [7], and show our proposed solution outperforms the existing methods even in...
harsh environments.

A. Outline

The outline of this paper is as follows. In section II we briefly introduce the extrinsic calibration model, and in section III we describe our proposed image processing solution. We show the point cloud processing procedures in section IV. We apply the cylindrical projection to convert the 3D sparse point cloud to 2D and perform dense image completion, converting the 3D sparse point cloud into a 2D dense image for easier feature extraction. We solve the occlusion effect of laser scans during the projection using foreground-background object identification and separation. We emphasize the importance of depth, reflectivity, and foreground object features while densifying the point cloud data. In section VII we formulate the calibration problem through a designed statistical objective function. Given multiple features and our objective function, we address the problems in current methods that degrade the calibration, which is the bias and numerous local optima in the objective function space. Section VII expresses the results of the Multi-FEAT pipeline. We also compare several different methods and their performance in single-frame scenarios. The final result of the work is validated based on the KITTI dataset. Discussions on targetless calibration and future work are presented in Section VII.

B. Related Works

In essence, the joint calibration problem of the camera and LiDAR is to estimate the relative pose between these two sensors, which is conventionally tackled with known external reference objects in the field. As early as 2004, the checkerboard was proposed as an external reference to calibrate the single-beam LiDAR and monocular camera [1], which was further improved by exploiting the plane-to-plane correspondence as an additional constraint [8]. In order to avoid moving the checkerboard repeatedly to create a multipose correspondence, multiple checkerboards were placed in the same field at different positions to create a connection between matching points and planes [9], which improved the then state of the art. As an alternative to checkerboards, various other design objects with different shapes have been proposed and experimented in the recent years e.g., rings [10], a trihedron [11], a sphere [3], V-shape objects [12] [5] and cuboids [13]. However, the reliance on artificial objects limits the feasibility of these solutions and confines their applicability to laboratories and test fields. An object-less or a targetless calibration solution will not only minimize human intervention to enable commercial applications such as autonomous vehicular systems [14], but will also enable autonomy in space-systems of the future [15].

In a targetless environment, known features (e.g., light poles in the vehicular environment) can be extracted using deep neural networks, e.g., PSP-Net [16] and BiSeNet-v2 [17], which can possibly be used for calibration. Furthermore, dedicated neural networks have been proposed for camera-LiDAR calibration, e.g., NN-based RegNet [18] and CalibNet [19]. However, these supervised neural network-based methods require large datasets for pre-training, and have relatively high computational power requirements, and their performance is restricted to the training datasets. We therefore limit our discussion to unsupervised optimization methods in this article.

A promising approach for targetless joint calibration of camera-LiDAR is to maximize the mutual information between the grayscale intensities of the camera image and the reflectivity values of the LiDAR [20]. An alternative to reflectivity is to use the depth discontinuity of the LiDAR output to capture the edge information of any object that is ubiquitous in the road environment [21], [22]. Taylor et al. used gradient orientation measurement to match the LiDAR reflectivity map with the camera grayscale image [23]. Another novel algorithm based on edge matching is the idea of solving the calibration-fusion joint problem [24], where the laser point cloud is projected onto the camera image plane and the projected sparse points are used for depth completion and match the edges of the depth map and the intensity image using cosine similarity.

In general, the existing calibration methods often use multiple frames or omnidirectional cameras to enlarge the amount of data measured, thus providing robust performance in estimating the calibration parameters. In this work, we propose a novel workflow Multi-FEAT, which relies on only one shot. It promotes and supplements the shortcomings of several current edge-alignment methods and uses multiple features to improve the reliability of the results.

C. Problem Formulation

A point cloud is a set of measurements from LiDAR sensors. We denote the LiDAR point cloud as $\psi$, and all the points that belong to the set of the point cloud are given by $\{p = [x, y, z] \in \mathbb{R}^3 | p \in \psi\}$. The camera image is a matrix of integers, denoted as $I \in \mathbb{Z}^{M \times N}$, where $M$ and $N$ are the dimensions of the image. We mark each pixel in the camera image as a 2D vector $u = [i, j]^T \in \mathbb{R}^2$, and thus the intensity of the pixel can be represented as $I_{i,j}$ at where $1 < i < M$ and $1 < j < N$. Typically, the camera image and the LiDAR point cloud are associated using the following rigid body transformation $[1]$

$$
\begin{bmatrix}
    u \\
    1
\end{bmatrix} = K \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix}
    p \\
    1
\end{bmatrix}
$$

where the camera intrinsic matrix $K$ matrix projects the objects from 3D to 2D space and thus forming an image. The matrix $\Omega$ determined by $R$ and $t$ contains the extrinsic calibration parameters, representing the rigid transformation from LiDAR coordinate system to the camera coordinate system and this transformation matrix belongs to the $SE(3)$ Lie group. Here, $R$ is known as 3D rotation matrix, determined by the rotation angles $r_x$, $r_y$, and $r_z$. The translation vector $t$ includes the 3 translational directions, $t_x$, $t_y$ and $t_z$.

The goal of edge alignment is trying to match the edges of one frame of the point cloud from LiDAR and one image from the camera and subsequently estimate the unknown
extrinsic calibration parameters $\theta = [r_x, r_y, r_z, t_x, t_y, t_z]^T$. In the following sections, we focus on the methods that extract the edge information from different modalities of sensors and the way to design and optimize factors that represent the extent of edge alignment, thus deriving the calibration parameters.

II. IMAGE PROCESSING

As both camera and LiDAR are perceptual sensors sharing a large FOV, their detection results in the environment have a considerable proportion of the overlapping area. Therefore, we aim to design a calibration algorithm exploiting the common features and aligning them. The metric of verifying this is achieved by the edges or boundaries of objects. One of the crucial concerns is that due to different modalities of sensors, cameras and LiDARs do not necessarily observe the same features, even if their perspectives are nearly identical. In this section, we focus on promoting edge features from camera images. Edges are located where pixels have distinct intensity differences and are usually extracted by gradient kernels. As stated in [21], the choice of gradient kernels is trivial since neither of the simple gradient-based kernels can extract exactly the desired information. Since images contain more delicate information than LiDAR, it is hard to decide which edges are excess for further edge alignment. In this work, we adopt the Sobel kernel, which can extract an edge map efficiently.

The mechanism of digital images determines that the edges on the camera images only reflect the difference in grayscale intensity values, which is heavily influenced by the light condition. Therefore, many boundaries that can be represented in the laser point cloud are not visible in the camera image because of the low grayscale difference or the effect of shadows. Under road conditions, complex roadside obstructions, such as street trees and billboards, can leave shades in the FOV under sunlight. These shadows are faithfully recorded by passive sensors such as cameras. When extracting edge information from an image, the shadiness leaves a fairly distinct boundary that is not present in LiDAR. Furthermore, the details in shaded regions will be ignored due to low contrast in high-exposure cases. On the other hand, this part of the boundary information is clear LiDAR, as it is an active sensor. The difference in the characteristics of edges between the two sensors will result in potential mismatches, affecting the performance of the calibration algorithm. It is hard for conventional methods to solve this problem entirely from the image side [25], but there are still ways to reduce the effect. Our approach uses simple yet effective histogram equalization for image enhancement to reduce the error in these scenarios. Consider the Sobel edge map of the original image as shown in Figure 3(a), and the effect of the proposed histogram equalization in Figure 3(b). Our method changes the grayscale histogram of the original image from its gray interval to a uniform distribution in all grayscale ranges, thus enhancing the local contrast and thus revealing the details in the shades, e.g., the shadow of trees and buildings.

Furthermore, to penalize the points that locate far from the edges and reward the points close to the edge on the intensity image, it is necessary to expand the current edges wider to cover more areas. In [21], the authors proposed an inverse distance kernel to reward pixels that locate close to edges. The closer to the edges, the higher the values. In our approach, we achieve the same goal simply by applying the Gaussian kernel on the edge map $E$ to expand the range of edges, leading the calibration parameters to move further toward the edges according to smooth transitions created by the Gaussian kernel. Note that the highest values are still assigned to the location of the original edge. Therefore, using a Gaussian kernel to smooth the edge map is adequate for the aim of edge expansion. The expanded edge map, denoted as $E_C$, will be used in section V for the Multi-FEAT optimization as illustrated in Figure I-B.

III. POINT CLOUD PROCESSING

In this section, we investigate the pre-processing of the LiDAR. In order to achieve the goal of alignment between the camera and LiDAR outputs, we need to perform an edge extraction on the point cloud, which is non-trivial since the points are arbitrarily scattered in 3D space with continuous values. In the Multi-FEAT pipeline, we apply a cylindrical projection on the point cloud, converting the 3D sparse point cloud into a 2D sparse image, i.e., we convert the 3D-2D (camera-LiDAR) calibration problem into a 2D-2D image registration problem (III-C). To acquire high quality edge
information from the 2D map, we design a workflow (see [I-B]) that resolves the occlusion effect (III-D). We further enrich the edge information by incorporating depth, reflectivity and foreground objects, generating high-resolution panoramic maps by solving the basis pursuit target function with total variation regularizer (IV-A). In the final stage of point cloud segmentation, we use the Canny edge detector to extract edge variation regularizer (IV-B) that resolves the occlusion effect (III-D). We further reveal the edge information by incorporating depth, reflectivity and foreground objects, generating high-resolution panoramic images visualized in (b), where the histogram equalization improves the local contrast on those underexposed regions. Finally, in (c), the image visualizes the idea of edge expansion using Gaussian filtering.

Fig. 3. Illustrations of image processing: The details in the Sobel edge map of the original image in (a) are not clearly visible, while those details are mostly revealed in (b), where the histogram equalization improves the local contrast on those underexposed regions. Finally, in (c), the image visualizes the idea of edge expansion using Gaussian filtering.

A. Plane segmentation

The 3D point cloud $\psi$ obtained from the LiDAR captures surroundings comprising both the foreground and background objects. However, the edges of these objects suffer from the occlusion effect since the projection plane of the LiDAR is typically non-parallel to the emission direction [26]. Furthermore, because the original LiDAR point cloud lies in a continuous 3D space, the projected panoramic map also suffers from spurious points which do not belong to an object of interest, e.g., ground points. This interference is seen as arcs in the Birds’ eye view (BEV) of the raw point cloud in Figure 4(a). In order to eliminate the interference from ground points and subsequently improve the identifiability of objects in the FOV, we use RANSAC (RANdom SAmple Consensus), a computationally efficient and robust iterative algorithm to eliminate outliers [27]. It is particularly efficient for extracting major geometry features, such as the ground plane in our case. We propose a RANSAC-based plane segmentation on the LiDAR point cloud $\psi$ for further removal.

In each iteration, three points in the FOV are randomly selected to construct a unique plane function, generalized as $Ax + By + Cz + D = 0$, with 4 parameters $A$, $B$, $C$ and $D$. With the plane model function, the distance of each point $p = [x, y, z]^T$ to the plane can be calculated by

$$d_p = \frac{|Ax + By + Cz + D|}{\sqrt{A^2 + B^2 + C^2}} \quad (2)$$

After obtaining the optimal plane model, we classify all the points within the point cloud, and denote the selected object points (or non-plane points) as

$$\psi_1 = \{ p \mid p \in \psi, \quad d_p > \gamma \} \quad (3)$$

where the constraint $\gamma$ that assigns points within the plane model is set as 0.2 meters, considering the non-flat ground. Figure 4 illustrate the performance of the RANSAC plane segmentation algorithm from the perspective of top view.

B. Foreground-Background objection classification

Given the processed LiDAR point cloud $\psi_1$ (see Figure 4(b)), we now aim to segregate the foreground and the background objects within the FOV. Density-Based Spatial Clustering of Applications with Noise or DBSCAN is a iterative density-based clustering algorithm that can identify densely connected regions. The plane segmented point cloud $\psi_1$ obtained from (3) is inputted to the DBSCAN algorithm, along with the hyper-parameters $R$ and $m$, which denote the radius of the detection ball and the minimum number of points $m$ within the $R$-ball to form a cluster respectively. The DBSCAN algorithm returns $K$ clusters in the FOV, and each point is assigned a cluster index $k \in \{1, \ldots, K\}$. To distinguish between the foreground and background objects, we aim to estimate the distance to the centroid of the $k$th cluster as

$$d_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \sqrt{x_i^2 + y_i^2} \quad (4)$$

where $k$ indicates the index of the cluster, $N_k$ is the total number of points in the $k$th cluster, and $\{x_i, y_i\}$ are the $i$th point in the $k$th cluster. Let $o_k \in \mathbb{N}$ denote the flag for the $k$th cluster indicating if the cluster is part of the foreground, background or an empty, then we have

$$o_k = \begin{cases} 2, & \text{if } d_k < \delta \text{ (foreground)} \\ 1, & \text{if } d_k > \delta \text{ (not foreground)} \\ 0, & \text{otherwise (empty)} \end{cases} \quad (5)$$

where $\delta$ is a known threshold in meters, and $o_k$ is set to 1, to indicate background, plane and other outliers. In summary, each point of the point cloud $\psi_1$ is associated with a cluster $k$, and each cluster $k$, along with all the points within the cluster, are categorized as either foreground, not foreground, or empty. The collection of only foreground points in denoted by $\psi_2$, which is shown in Figure 4(c) in red.
C. Cylindrical projection

We now apply a cylindrical projection on the point cloud, converting the 3D sparse point cloud into a 2D sparse image. It is a non-linear conversion from 3D continuous coordinates on a 2D discretized plane. LiDAR points are projected on a cylindrical plane and assigned to the nearest cells. In other words, cylindrical projection finds the corresponding 2D coordinates of each point in the 3D point cloud. It allows processing the 3D point cloud in a 2D form with the chosen feature channel. Given a point \( p = [x, y, z]^\top \) in a 3D point cloud, the corresponding cylindrical projection in 2D is given by

\[
\begin{bmatrix}
  i \\
  j
\end{bmatrix} = \begin{bmatrix}
  \frac{h}{\delta_h} \arctan \frac{y}{x} \\
  \frac{h}{\delta_v} \arctan \frac{z}{\sqrt{x^2 + y^2}}
\end{bmatrix}
\]  

(6)

where the vector \([i, j]^\top\) correspond to the 2D coordinates on the panorama and \( h \) is the scaling parameter, determining the size of the image. The horizontal and vertical scanning resolutions are denoted by \( \delta_h \) and \( \delta_v \) respectively, which are calculated from the sampling frequency of the LiDAR. Since all the points in 3D carry spatial information, we apply the cylindrical projection to the whole point cloud \( \psi \).

D. Occlusion deduction

After successfully classifying the foreground and background points given the flag \( o_k \), we can perform occlusion deduction at the image level in 2D. To ensure the background objects do not blend with foreground objects, we design a morphological filter and a masking trick to separate foreground and background pixels in 2D. We use morphological operators [25] to design a foreground mask \( D \) to highlight the foreground objects. If background points are blended in this region, they are marked invalid since the foreground targets should occlude the background objects from the viewer’s perspective.

We first apply the dilation to fill the pixel gaps and then erosion to reduce the growing size of object regions. Finally, we denote the panorama only with foreground pixels as \( u_{obj} \). Dilation for the grayscale image is to apply a statistics filter that returns the local maximum intensity value within the range of its neighbors defined by a sliding kernel. We perform the dilation to the sparse foreground panorama. Therefore the tiny pixel holes within LiDAR realization areas can be connected as a mask. However, such a mask is enlarged from its original territory due to the size of the dilation kernel. To diminish this influence, we perform the erosion right after the dilation with the same kernel size.

Since the pixel gaps inside the object are mainly filled while the enlarged area is shrunk to the original size, the sparse foreground region is densified as a foreground mask \( D \). All
the background points with label \( o = 1 \) while located on the foreground mask are discarded directly from the original point cloud \( \psi \). Therefore the points that could cause the occlusion effect no longer exist. After these steps of morphological operation, the results of the dense depth map are improved on the edges, as shown in Fig 5. The whole point cloud preprocessing is shown in Algorithm 1.

![Algorithm 1: The occlusion deduction algorithm.](image)

Fig. 6. The dense maps of depth, reflectivity, and object features, were recovered by minimizing the constrained total variation norm. The depth map is smooth and continuous in major areas, while the reflectivity map is noisy. However, the high reflectivity areas are distinct and easy to extract. The object map loads the artificial features, providing the foreground features given the clustering results. It enhances the boundaries of different targets that are hard to recognize, given the depth map.

The LiDAR information is sparse and cluttered. Projecting the LiDAR information on the sparse image can be analyzed as ordinary images. The recovery of the dense map can be formulated as a basis pursuit problem given the sparse image signal, considering the sparse image as a downsampled from the original image.

When dealing with the panorama after cylindrical projection, we are interested in the features carried by the map. In the previous section, we applied the foreground feature \( o = 2 \) to perform morphological filtering. In order to exploit the information from LiDAR, we extract three feature channels as depth, reflectivity and foreground features, feeding them to the following dense map completion process three times to generate panoramas with three different features.

Depth information is widely used as it is an essential feature benefiting from using LiDAR. It compresses the 3D spatial information as a single set of values - the distance of each point to the sensor. It reveals the geometric relationship between objects. Therefore the depth gap tends to be the physical edge that is usually seen from camera images. The second feature we exploit is laser reflectivity, carried directly by the LiDAR measurements. It is less mentioned in other edge alignment-based calibration methods since the measurements sometimes are unreliable due to the noise [28]. Although the majority of the reflectivity values are low, and little information can be derived from such regions, in the road scenario, many car plates, traffic cones, signs, and other traffic-related objects use illuminating materials providing extraordinarily high reflectivity values. Also, white areas and vegetation generally provide higher reflectivity values. Therefore, the reflectivity values can be used to add some details to the surface patterns that the depth feature neglected. The third feature is the foreground objects we previously extracted. It enhances the boundaries of foreground objects, providing the bottom edges that are not distinct in the depth map. The edges of the objects are ideally visible in the camera image, which is crucial for edge alignment. In Fig 6 we give examples of the dense maps of these three feature channels. The sparse depth map, reflectivity map and foreground map are denoted as \( u_d \), \( u_r \) and \( u_o \), respectively.

Suppose \( \phi \in \mathbb{R}^{N_x \times N_y} \) denote the desired dense panoramic image, where \( N_x \) and \( N_y \) indicate the dimensions of the image along the \( x \) and \( y \) dimensions respectively, \( u_L \) represents the 2D sparse panoramic image obtained from 3D point cloud, and \( H \) the known binary masking matrix, which is created based on the locations on the panorama with LiDAR projection. We then have

\[
\phi = \arg\min_{\phi \in \mathbb{R}^{N_x \times N_y}} \left\{ \| u_L - H \odot \phi \|_2^2 + \lambda \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} (\| \nabla_x \phi \|_1 + \| \nabla_y \phi \|_1) \right\}
\]

(7)

where \( \odot \) is the element-wise matrix multiplication. We formulate the data fidelity within an \( \ell_2 \) norm that constrains the feature values in the given positions by the masking matrix. The rest terms accumulate the total gradient in both \( x \) and \( y \) direction of the desired map, as shown in (7). By minimizing the unnecessary and rapid gradient changes due to noise or sparse information, we obtain smooth changes in the whole

IV. MULTI-FEATURE EDGE EXTRACTION

A. Dense Map Completion

The LiDAR information is sparse and cluttered. Projecting all the 3D points on the cylindrical plane only solves the latter problem. It is necessary to construct dense maps so that the LiDAR information on the sparse image can be analyzed
image, while the sharp edges are kept sharp since the \(\ell_1\) norms promote sparsity on the solution.

The whole problem is formulated as a variant of the ROF (Rudin-Osher-Fatemi) image restoration model \([29]\). The dense panoramic map can be recovered by minimizing the data fidelity loss and total variation norms. The whole problem can be optimized with FISTA or Split-Bregman algorithms in very few iterations \([30]\) \([31]\). We apply an open-source toolbox from \([32]\) to solve these optimization problems.

### B. Point Cloud Edge Extraction

In this section, we aim to blend the different extracted features into a single edge map. We begin with an effective edge extraction using the Canny edge detection \([33]\), which includes four main steps: Gaussian filtering, Sobel gradient and orientation, non-maximum suppression, and hysteresis thresholding. The output edge maps are \(E_d\), \(E_r\), and \(E_o\), denote the depth image, intensity image and object image respectively.

We define the mixed edge map \(E_L\) as the average value of three edge maps after the Canny edge detection, shown in \([8]\), where we equally weigh each of the features, as follows

\[
P(E_L(i,j)) = \frac{1}{3} \sum_{k \in \{d,r,o\}} E_k(i,j), \tag{8}
\]

where \(P(.)\) denotes the edge probability, implying the probability of a pixel on the edge map being an edge. The mixed map provides more details, bringing rich edge information for further alignment, as shown in Figure 7

### V. Objective Function and Optimization

This section introduces the design of the objective function with LiDAR multi-feature edge map and camera edge map.

#### A. Design of the Objective Function

As explained in previous sections, the edges of the camera image are not originated from the same source compared to the edges of the LiDAR panorama. For example, the alignment of edges of shadows is complex because of unmatched features. Such mismatch may result in a local optimum on the objective function. In order to avoid mismatch, we adopt multiple pre-processing steps for both the camera image and LiDAR point cloud. Although the topology of camera image edges looks similar to the multi-feature edge map, applying traditional image registration methods is not easy. The scale invariance and the unknown affinity transform parameters do not directly associate with extrinsic calibration parameters. Therefore, we project the edge points identified from multi-feature cues onto the camera image plane, maximizing the linear similarity defined by the inner product of the corresponding image edge intensity values and the multi-feature edge probabilities. If all edge points matched perfectly on the intensity edge map after projection, the final value should be at the local maximum.

We utilize a normalizing factor to indicate the percentage of "correctly matched" points. Assume there are in total \(N\) points in the point cloud. Among them, \(N_e\) points are edge points selected with non-zero edge probability. If an edge point is projected onto the intensity pixel edge of the camera image, the counter \(N_m\), which stands for the number of matched points, will accumulate. This ratio is mathematically defined as \(\frac{N_m}{N_e}\), which can also be interpreted as precision. Ideally, the correct calibration parameters will result in the highest ratio of \(\frac{N_m}{N_e}\). By coarsely binarizing edge/non-edge to intensity image, the influence of the actual value of the camera edge is further decreased.

We design the cost function by constructing linear dependence between edge intensities of camera images and edge probability acquired from multi-features LiDAR point clouds. It is determined as the sum of the multiplication of the intensity of the grayscale edge, which \(E_C\) denotes, and the probability of being an object edge in the point cloud, defined earlier as \(P(E_L)\).

\[
J(\theta) = \frac{N_m}{N_e} \sum_{n=1}^{N} E_C(\theta)_n \cdot P(E_L(i_n,j_n)) \tag{9}
\]

The \(E_C(\theta)_n\) is the corresponding edge value of on the grayscale image of the projected point \(p_n\) given an extrinsic parameter \(\theta\) according to \([1]\). The edge probability of the \(n\)-th point \(p_n\) is \(P(E_L(i_n,j_n))\), calculated by mixed edge map. The strongest edge present on all three maps has the highest weight, while the non-edge points do not participate in the process, providing less consumption of computational power.

#### B. Optimization

To successfully solve the target function \([9]\), common choices include the gradient-based or Hessian-based optimisers. In our case, the choice is not particularly important since we have already added features to promote the concavity of the function. In our pipeline, we choose the Barzilai and Borwein gradient ascent method \([34]\). The explicit expression of the gradient of \([9]\) is
\[
G = \nabla J(\theta) = \frac{J(\theta + \Delta h) - J(\theta - \Delta h)}{2 \cdot \Delta h},
\]
and the numerical derivatives can be calculated for a small parameter \(\Delta h\). The update equation of the gradient ascent method is defined as
\[
\theta_{k+1} = \theta_k + \gamma_k \frac{G_k}{\|G_k\|}
\]
where \(k\) indicates the \(k\)-th step in the iteration. The adaptive step size is also defined as
\[
\gamma_k = \frac{s_k^T s_k}{s_k^T g_k}
\]
where \(s_k = \theta_k - \theta_{k-1}\) and \(g_k = G_k - G_{k-1}\). The proposed Multi-FEAT algorithms is summarized in Algorithm 2

**Algorithm 2:** Multi-FEAT optimization pipeline

**Input:** Multi-feature edge map \(E\), Initial extrinsic parameter \(\theta_0\), Camera image \(I\), Error \(\epsilon\), \(k=0\)

**Output:** The estimated \(\hat{\theta}\)

1. while \(\|\theta_{k+1} - \theta_k\| \geq \epsilon\) do
   2. Calculate \(J(\theta_{k+1})\)
   3. Calculate \(G_k\)
   4. Calculate \(\gamma_k\)
   5. Update \(\theta_{k+1} \leftarrow \theta_k\)
   6. \(k = k+1\)
   7. end
8. \(\hat{\theta} \leftarrow \theta_k\)

**Hyper-parameters**

| Parameter | Values |
|-----------|--------|
| the threshold \(\gamma\) for RANSAC in (3) | 0.2 |
| the minimum points \(m\) to form a cluster | 8 |
| the range \(R\) to form a cluster in DBSCAN | 0.25 |
| the multiplier \(\lambda\) of TV norm in (7) | 0.05 |

**TABLE I**

The hyper-parameters used in the Multi-FEAT algorithm.

**VI. SIMULATIONS**

We now evaluate the performance of the proposed Multi-FEAT pipeline based on the open-source KITTI dataset [35]. The KITTI dataset comprises data collected from a high-resolution image camera and a Velodyne laser scanner onboard a standard station wagon, driving around the city of Karlsruhe in rural areas and on highways. In addition to the raw data, the ground truth values of the calibration parameters are also available from this dataset.

Furthermore we compare the objective function of our approach with that of existing targetless solutions, Pandey et al. [20], Levinson et al. [21] and Castorena et al. [24], respectively. The shapes of objective functions with rotation and translation displacements are presented. For rotational parameters, we use the interval \(-0.3\) to \(0.3\) rads, and for the translation parameters \(-0.3\) to \(0.3\) meters. Note that the objective function in Castorena et al. needs minimization, while the rests need maximization. For the sake of clarity, we normalize the cost functions into interval \([0, 1]\). A summary of all the hyper-parameters used in the Multi-FEAT optimization problem is summarized in Table I

**A. Single-frame evaluation**

To analyze the effectiveness of our proposed Multi-FEAT pipeline, we select unique urban scenarios within the scope of the KITTI dataset.

1) Scenario 1: Urban Road: In this scenario, shown in Fig. 8, pedestrians are crossing the road, the vehicle in the middle of FOV, the cyclist on the road, the tram rail, the traffic signs, and buildings in the background.

We notice in Fig. 8, that Pandey et al.'s method in the first column provides the local maxima around the ground truth values for almost all the six parameters. It is generally non-smooth for this single-frame trial. The global optimum is more distinct in the method by Levinson et al.. However, various local maxima would impede the gradient-based optimizers from achieving the true value. Castorena et al.’s method also has some limitations since it overlooks the occlusion effect, and it is evident that the shape of the cost function is not smooth. It is solved by a simulated annealing-based method that might reduce the influence of local optima at the cost of more iterations. However, the global optimum is not located in the center of the curves, which will bias and hence...
Fig. 9. Scenario 2 (Highway) presents an example of highway scenario. LiDAR cannot see through the near-field vehicles around, therefore providing little information from farther objects.

2) Scenario 2: Highway: The second example is from the main road (highway) with vehicles around. There are clear lanes and traffic signs in the distance. The vehicle on the left is too near to the on-board LiDAR, and hence many laser beams that should illuminate the vehicle’s body are reflected elsewhere that the LiDAR cannot receive. The densely packed vehicles in the periphery make clustering and separating objects challenging. In Fig 9, we show the image with laser points and the results.

In these complex scenarios with closely spaced objects under a heavy shadow, the performance of Pandey et al.’s method is no longer robust. For $t_x$ and $t_z$, the mutual information assumption does not hold in this scenario. We observe numerous local optima in case of the methods proposed by Levinson et al. and Castorena et al.. In contrast, our proposed approach shows a smooth objective function for the estimation of the rotation parameters. However, estimating the translation parameters may lead to a minor bias.

3) Scenario 3: Suburban Area: The third experiment covers a complex environment in a suburban area. The experiment vehicle, with the on-board camera and LiDAR, faces a ramp with a positive inclination. Given this view, observe that the plane segmentation cannot extract both the flat ground and the ramp. Furthermore, some plane points participate in the object clustering, hence degrading the results. In the image, we notice the motorbike rider is coming down the ramp, and is completely hidden under the shade of the trees. In general, more than half of the image is corrupted by the shadow of the trees. A wire pole stands tall to the left of the image, but is close to the wall, with little depth discontinuity information to be exploited. The clustering fails to separate the motorbike rider and the wire pole effectively. Hence, in the FOV of the experimental vehicle, there are few valid objects for edge alignment. The results are illustrated in Fig 10.

Pandey et al.’s approach faces severe challenges since there is a weak correlation between image intensity and laser reflectivity, primarily due to the shadows. The other edge alignment methods need valid objects in the FOV, which is severely limited in this scenario. It is clear that none of the existing targetless methods offer a smooth objective function for any of the 6 parameters. The performance of our proposed method is also severely affected in this complex environment, however the rotation parameter $R_y$ and translation parameters $\{t_x, t_y\}$ show promising results with a clear maximum.
Fig. 11. The box plots show the error residuals \( \{ \epsilon_i \}_{i=1}^{30} \) of calibration parameters estimated using the Multi-FEAT algorithm over 30 frames, where the true calibration parameters are constant.

### B. Multi-frame evaluation

In the previous sections, we investigated the performance of the proposed Multi-FEAT against using various single-frame scenarios of the KITTI dataset. In this section, we arbitrarily select multiple frames during the same test round. More concretely, we select 30 frames during which the 6 parameters are constant and provided in the KITTI dataset. These true values of the 3 rotation parameters and the 3 translation parameters are denoted by \( \theta \), indicated in the first row of Table II.

To understand the performance of our proposed solution over multiple frames where the parameters are expected to be constant, we estimate these parameters individually for 30 frames using the Multi-FEAT algorithm, and compute the mean value of over these multiple frames. Let \( \hat{\theta}_i \) denote the 6 unknown parameters computed using the Multi-FEAT algorithm at the \( i \)th frame, then the proposed estimate \( \hat{\theta} \) over the 30 frames is given by

\[
\hat{\theta} = \frac{1}{30} \sum_{i=1}^{30} \hat{\theta}_i
\]

(13)

Now, to compare our proposed estimator with the true value given by the KITTI dataset \( (\theta) \), we use the MAE (Mean absolute error) to minimize the dependence on outliers i.e.,

\[
\text{MAE}(\hat{\theta}, \theta) = \frac{1}{30} \sum_{i=1}^{30} \epsilon_i = \frac{1}{30} \sum_{i=1}^{30} (\hat{\theta}_i - \theta)
\]

(14)

where \( \hat{\theta}_i \) is the proposed estimator using Multi-FEAT \( (\hat{\theta}) \) and \( \epsilon_i \) is the error residual, which we plot in Figure 11 for all the estimated rotation and translation parameters. The mean of the values indicate the MAE, which is closely centered around 0 for the rotation parameters. However, in Figure 11(b), we observe a larger dispersion of translation errors along the \( y \) and \( z \).

For each calibration parameter, we calculate the mean value of 30 single-frame estimations \( (\hat{\theta}) \) and the corresponding MAE \( \text{MAE}(\hat{\theta}, \theta) \), which are tabulated in Table II. In the case of rotation parameters, the MAE (mean absolute error) of calibration results over 30 frames is lower than 0.021 rad (1.2 degree), which is sufficiently low. We do notice that the results on translation parameters tend to have higher biases, leading to poorer results. One of the reasons could be the displacement of the camera and LiDAR sensors, as they are not on the same horizontal level. Therefore, the bias is actually caused by the assumption of edge alignment since the perspectives of these two sensors are slightly different.

### VII. Conclusion

In this paper, we proposed a new multi-feature edge alignment algorithm using one-shot information for targetless extrinsic calibration between the monocular camera and LiDAR sensors. Multi-FEAT uses cylindrical projection to convert the 3D sparse point cloud into 2D images. We design a filtering module to solve the occlusion effect by clustering and morphological closing. By combining the depth, reflectivity, and foreground object edge features and formulating a statistical model, we enriched the information of the LiDAR point cloud, thereby obtaining a better fit with our designed objective function. We illustrated the mismatch and inertial bias by presenting the shape of the cost function for the 6-DOF calibration parameters. We perform extensive simulations to compare our proposed Multi-FEAT algorithm with several state-of-the-art targetless calibration methods using the KITTI dataset, which shows the advantages of our approach in diverse scenarios.

In the future, we aim to extend the proposed one-shot Multi-FEAT algorithm to multi-frame scenarios, similar to ideas presented in Section VI, however exploiting robust Bayesian techniques. Meanwhile, the adaptability of the calibration algorithms to the actual autonomous perception workflow is also a critical issue to be investigated. Hence, we would explore the ability of the online calibration algorithms to adopt some middle-level results from the perception tasks, such as results of semantic segmentation from camera images. Finally, we aim to investigate and reduce the computational complexity of the Multi-FEAT algorithm, making them more suitable for automotive applications.

### References

[1] Q. 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.

[2] H. Alismail, D. Baker, and B. Browning, “Automatic calibration of a range sensor and camera system,” 10 2012.

[3] M. Pereira, D. Silva, V. Santos, and P. Dias, “Self calibration of multiple lidars and cameras on autonomous vehicles,” Robotics and Autonomous Systems, vol. 83, 05 2016.

[4] 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.

| Parameter | \( r_1 \) (rad) | \( r_2 \) (rad) | \( r_3 \) (rad) | \( t_x \) (m) | \( t_y \) (m) | \( t_z \) (m) |
|-----------|----------------|----------------|----------------|-------------|-------------|-------------|
| \( \theta \) | 0.470 | -1.554 | 1.100 | 0.004 | -0.076 | -0.272 |
| \( \hat{\theta} \) | 0.482 | -1.576 | 1.113 | -0.020 | -0.150 | -0.213 |
| MAE | 0.012 | 0.021 | 0.013 | 0.016 | 0.073 | 0.059 |

Table II: Results from the multi-frame evaluation of the Multi-FEAT algorithm which show the true \( \theta \), estimated \( \hat{\theta} \), and mean absolute error (MAE) of the calibration parameters.
