IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 10, 2020, accepted July 26, 2020, date of publication August 4, 2020, date of current version August 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3014121

Targetless Camera-LiDAR Calibration in Unstructured Environments

MIGUEL ÁNGEL MUÑOZ-BAÑÓN, FRANCISCO A. CANDELAS, AND FERNANDO TORRES, (Senior Member, IEEE)
Automatics, Robotics, and Computer Vision Group (AUROVA), University of Alicante, 03690 Alicante, Spain

Corresponding author: Miguel Ángel Muñoz-Bañón (miguelangel.munoz@ua.es)

This work was supported by the Regional Valencian Community Government and the European Regional Development Fund (ERDF) through the grants ACIF/2019/088 and AICO/2019/020.

ABSTRACT The camera-Lidar sensor fusion plays an important role in autonomous navigation research. Nowadays, the automatic calibration of these sensors remains a significant challenge in mobile robotics. In this article, we present a novel calibration method that achieves an accurate six-degree-of-freedom (6-DOF) rigid-body transformation estimation (aka extrinsic parameters) between the camera and LiDAR sensors. This method consists of a novel co-registration approach that uses local edge features in arbitrary environments to get 3D-to-2D errors between the data of both, camera and LiDAR. Once we have 3D-to-2D errors, we estimate the relative transform, i.e., the extrinsic parameters, that minimizes these errors. In order to find the best transform solution, we use the perspective-three-point (P3P) algorithm. To refine the final calibration, we use a Kalman Filter, which gives the system high stability against noise disturbances. The presented method does not require, in any case, an artificial target, or a structured environment, and therefore, it is a target-less calibration. Furthermore, the method we present in this article does not require to achieve a dense point cloud, which holds the advantage of not needing a scan accumulation. To test our approach, we use the state-of-the-art Kitti dataset, taking the calibration provided by the dataset as the ground truth. In this way, we achieve accuracy results, and we demonstrate the robustness of the system against very noisy observations.

INDEX TERMS Camera-LiDAR, extrinsic calibration, target-less calibration, sensor fusion, mobile robots.

I. INTRODUCTION

Environment perception is an essential stage of autonomous navigation systems [1]. A large number of works focused on perception tasks, such as semantic segmentation [2], object tracking [3], or Simultaneous Localization And Mapping (SLAM) [4], [5], use both LiDAR and cameras to capture the environment information. Due to the sensors complementarity, it is interesting to combine both sources. To fuse the camera and the LiDAR data, we need to calibrate a six-degree-of-freedom (6-DOF) rigid-body transformation, aka extrinsic parameters, that relates both sensors. The accurate extrinsic parameter estimation is a current challenge in mobile robotics, because, due to the multi-modal nature of these sensors, the representation space of the data is significantly dissimilar, which makes it very hard to co-register information between both.

The most widespread methodology for camera-LiDAR extrinsic calibration is the target-based. In this case, known artificial targets are usually used, like a chessboard [6] or any other object with known geometry, and observable by both sensors. In this way, it is possible to estimate the rigid-body transformation between the two sensors by detecting the same object from the two viewpoints. However, this approach has two main drawbacks: First, it is very time-consuming in terms of the experimental process. And, second, due to the lack of known artificial targets, it assumes that the calibration is static while the robots are navigating in the environment. This assumption can be applicable in indoor navigation cases, where a mobile robot usually circulates on very smooth ground, and, thus, the sensors do not suffer significant disturbances that may lead to miscalibration. However, in outdoor environments, navigation conditions are often more adverse, and, due to uneven terrain, the calibration may vary over time. In this case, it would be interesting to have an algorithm that calculates the extrinsic parameters automatically and, therefore, without the need for known artificial targets [7]. In the AUROVA group at the University of Alicante, we are developing a research line in autonomous navigation in unstructured outdoor environments [8], [9]; for this reason,

The associate editor coordinating the review of this manuscript and approving it for publication was Shaohua Wan.

we have focused our current work on the last-mentioned calibration approach.

In this article, we present a novel automatic calibration method of camera-LiDAR extrinsic parameters. This method requires neither an artificial target (target-less calibration) nor an environment structuration. The new approach proposed is based on the co-registration of the points that present discontinuity features in the LiDAR measurement space with the points that describe the intensity of edges in the image. We performed the co-registration of points with a single laser scan, which allows applying the calibration method independent of any localization system. Then, we use a combination of Perspective-three-Points (P3P) and M-estimator sample consensus (MSAC) to estimate the relative transform that minimizes 3D-to-2D errors generated in the co-registration stage. With this method, the parameter estimation depends mainly on the number of points reliably co-registered. For this reason, and due to the high noise component found in unstructured outdoor environments, we consider each parameter estimation as a noisy observation in a dynamic system. These observations will update the state (current calibration) of a Kalman filter, depending on their reliability. The concrete contributions of this work are listed below:

- Our main contribution is a target-less calibration method for unstructured and arbitraries outdoor environments, which does not require, in any case, a localization system to accumulate laser scans in a dense point cloud.
- The presented method includes a novel algorithm for camera-LiDAR points co-registration. Our algorithm is based on maximizing the alignment of local features. On one hand, discontinuities features in the LiDAR points, and on the other hand, edge features in the image. We perform the alignment by search the best (in terms of an objective function) translation, rotation, and scale, of LiDAR local feature points.
- We use the Kalman filter to dynamically update the current calibration, taking the calibrations estimated by the proposed method as observations. Besides, we provide a methodology to characterize the observation's noise.

The outline of the rest of this article is as follows: In Section II, we review the previous main work on the subject, with particular attention to those focused on target-less calibration. We explain in detail the proposed method in this work in Section III, with a high-level division into three differentiated parts. In Section IV, we describe the filter modeled to refine the final calibration. We show experimental results in Section V, including filter noise modeling, as well as quantitative results against ground-truth. Finally, in Section VI, we present the main conclusions obtained from this work.

II. RELATED WORK

A large number of camera-LiDAR extrinsic calibration works are target-based. With this methodology, a known artificial target is used to be detected by both sensors. In [10]–[14], the authors use a chessboard as a target. Planar marks are also used in [15], but in this case, the drawing pattern presented is different. Other types of marks with different geometric shapes are also often used, such as spheres [16], triangles [17] or polygons [18], [19]. Cost functions are usually used to estimate extrinsic parameters, which minimize reprojection errors between the target detected with the LiDAR and projected on the image plane, and the target detected on the image itself. In [20], the authors use a function based on least squares. In other cases, such [21], the relative transformation between the current camera position and the desired position is used employing a 3D-to-2D motion transformation based on Perspective-n-Points (PnP) algorithms [22].

Currently, the main challenges in mobile robotics are autonomous navigation and scene understanding in outdoor environments. These conditions often cause unfavorable situations for extrinsic calibrations. For this reason, there are currently different works focused on target-less calibration methods. Within this methodology, we can distinguish two categories: motion-based and feature-based calibrations. Motion-based calibrations have the advantage of not needing any initial guess for method convergence, although they are usually less accurate. In contrast, feature-based calibrations need an initial guess for the method convergence, although, in this case, we generally get more accurate results.

Motion-based calibration methods estimate odometry with each sensor separately, and simultaneously estimate extrinsic parameters that minimize the error between calculated paths [23]–[25]. In [26], the authors estimate the parameters by solving the homogeneous transform equations $\mathbf{AX} = \mathbf{XB}$, where $(\mathbf{A}, \mathbf{B})$ are the estimated trajectories with each sensor, and $\mathbf{X}$ are the extrinsic parameters to calibrate. This type of calibration has the advantage that it does not need an initial guess for the algorithm convergence. However, the final results are often imprecise due to scale problems in visual odometry. In [27] the authors use a camera-motion estimation as a refinement stage in motion-based calibration. In the recent paper [28], the authors present a hybrid method that requires a motion-based calibration to obtain an initial guess and then adjusts the parameters more precisely using feature-based calibration.

As mentioned above, feature-based calibration needs an initial guess for convergence. However, we consider that this is not a very hard requirement since, usually, the approximate measurements of the components, where we mount the sensors on the robot, are available. Besides, we can make a software hand-fitting adjustment, leaving the fine-tuning to calibration algorithm. Some works in this category use global features from the image and the laser data in order to maximize an objective function (global-feature-based). In a seminal work [7], the authors use an objective function that maximizes alignment between the edges in the image and the discontinuities in the LiDAR scans. There are other different works in this line focused on improving the arrangement of the edges [29]–[31]. For example, in the recent work [32], the authors also use a cost function based on Gaussian Mixture Models (GMM) to maximize edge alignment. Another global-feature-based approach use an objective function that

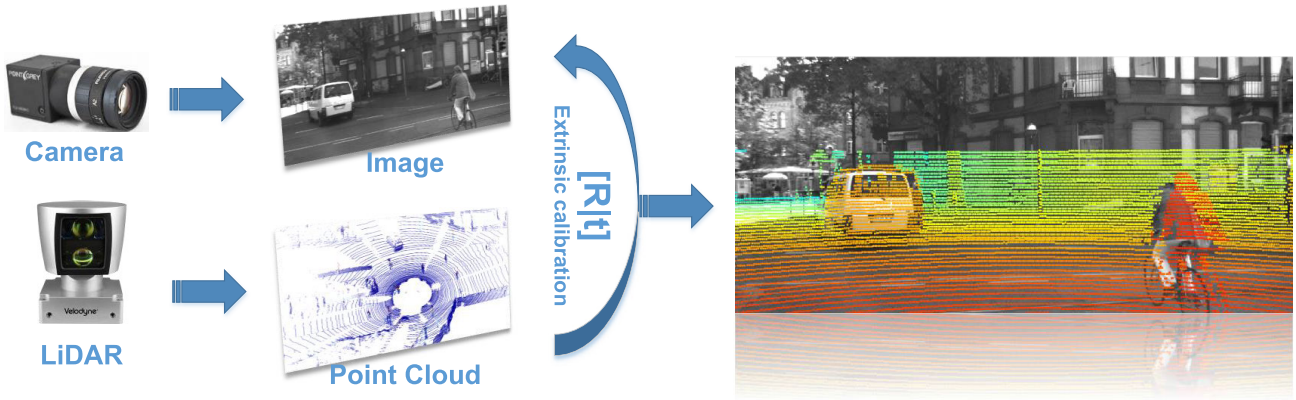

FIGURE 1. Example of camera-LiDAR sensor fusion. Each point measured in 3D space with the LiDAR is projected onto the image plane using the extrinsic parameters [R|t]. In the representation on the right side, we show an example of a camera image in grayscale, where we can see the projected LiDAR points in color. The objects measured with the LiDAR and with the camera, respectively (for example, the cyclist) do not coincide point by point in the image plane. This misalignment is an example of the miscalibration effect. The parameters [R|t] that align these objects in the image plane as precisely as possible will be what we will consider as the system calibration.

maximizes the Mutual Information (MI) between the intensity of the LiDAR measurements and the intensity of luminance in image [33]. From this seminal work, other variants arise to improve the primary method, e.g., in [34], the authors use a Bagged Least-Squares Mutual Information (BLSMI) for smoothing objective function to avoid local minimums in highly noisily outdoors environments. In [28], the authors use this method for fine-tuning after calculating an initial guess using motion-based calibration.

Local-feature-based is another approach for calibration, which first performs a co-register of LiDAR features with image features, and then minimizes reprojection errors to estimate extrinsic parameters. This approach has the advantage of being able to detect inliers during feature point co-registration. Therefore, a frame with image regions that are not appropriate to the co-registration may contain local information that is useful in other areas of that image, in opposite to the global-feature-based approach, in which an unfavorable local region in image may generate an imprecise parameter estimation. However, local-feature-based has the disadvantage that it is tough to implement without known targets. Many of the research works that perform it use geometric primitives, like planes or straight lines, to co-register LiDAR points with points in the image. In [35], the authors use road structuring in urban areas, which would limit calibration to these kinds of environments. Instead, in [36], [37], the authors use the co-registration of modeled straight lines in both spaces. In these cases, the problem is that the environment must be highly structured to contain enough geometric primitives.

In this article, we present a method focused on a local-feature-based calibration approach by co-registering edge feature points between LiDAR and camera data. In this case, we don't require geometric primitives or any other kind of environment structuring. Furthermore, our method does not require to accumulate LiDAR scans to obtain a denser representation of the environment. This fact involves the great advantage of not needing any localization system, which would introduce additional complexity to the complete system. However, this does not mean that the method necessarily works in a single scan, since we can use the information from different time points to accumulate co-registration errors, or for calibration filtering. For the extrinsic parameter estimation, we use a combination of P3P (algorithm based on PnP [38], [39]) and MSAC [40], which allows detecting the number of inliers in matches. This information enables quantifying the noise in the estimated calibration. Thanks to this noise quantification, we implement a Kalman filter [41] to update the current calibration depending on the measurement reliability.

III. TARGETLESS EXTRINSIC CALIBRATION METHOD

For camera-LiDAR sensor fusion, it is necessary to project each point measured by the LiDAR on the image plane defined by the camera intrinsic parameters. We show below (1) the expression to transform from 3D coordinates in the LiDAR frame system to 2D coordinates in the image plane.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \left[\mathbf{R}|\mathbf{t}\right]^{CL} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}^{L} \tag{1}$$

where $\mathbf{K}$ is the intrinsic parameters of the camera [42], $\left[\mathbf{R}|\mathbf{t}\right]^{CL}$ is the six-degree-of-freedom (6-DOF) rigid-body transformation, $\mathbf{t} = (t_x, t_y, t_z)$ is the translation vector, and $\mathbf{R} = f(\theta_x, \theta_y, \theta_z)$ is the rotation matrix. The superscripts in (1) indicate the frame in which the variables are expressed, where C represents the camera frame, L represents the LiDAR frame, and CL represents the LiDAR to camera frame transition. We show an example of LiDAR points projection on the camera image plane in Fig. 1. In this example, given a poorly calibration, the same objects captured by both sensors do not coincide in the image plane. Then, we can define the problem as the estimation of $\left[\mathbf{R}|\mathbf{t}\right]^{CL}$ that allows the objects

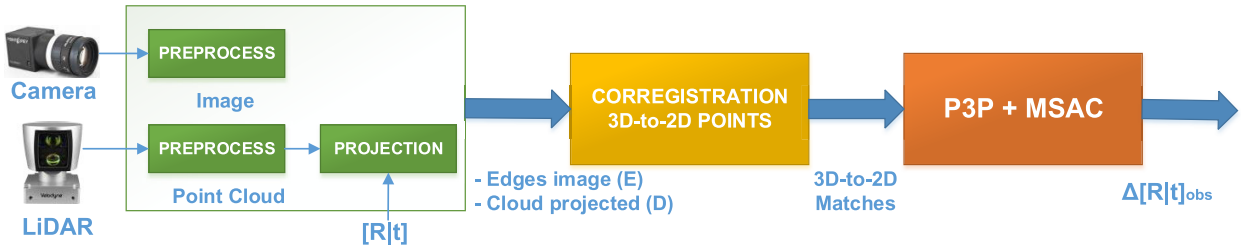

FIGURE 2. We can divide the proposed calibration method into three stages. In the first stage, we perform the data preprocessing of each sensor and the projection of the LiDAR points in the image plane. The second stage receives the processed data and co-register the LiDAR feature points with points from the features in the image, thus obtaining the 3D-to-2D matches and its errors. In the third stage, we estimate extrinsic parameters using the PnP-based P3P algorithm and perform inlier detection using MSAC, which is a variant of RANSAC.

captured in the environment to be aligned as accurately as possible. From now, for the sake of clarity, we define the extrinsic parameter vector as $\mathbf{\Theta}^{CL} = (t_x, t_y, t_z, \theta_x, \theta_y, \theta_z)$. Where $\left[\mathbf{R}|\mathbf{t}\right]^{CL} = f(\mathbf{\Theta}^{CL})$. Like any feature-based method, we consider that we have an $\mathbf{\Theta}_0^{CL}$, aka initial guess. In this way, poor initialization can potentially affect the algorithm convergence. However, it is usually not complicated to provide an initial guess that favors convergence by using the physical measurements of the robot or by using a software hand-fitting.

We divide our method into three clearly defined stages (Fig. 2). In the first stage, we preprocess the data to obtain the edges features in the image, and discontinuities features in the LiDAR point cloud (Section III-A). In the second one, we co-register the edge features between both spaces, employing edge alignment maximization using separate clusters, i.e., at object level (Section III-B). Finally, in the last stage, we estimate the relative motion transform, from which we can infer $\hat{\mathbf{\Theta}}^{CL}$, which minimizes the 3D-to-2D error obtained in the previous one, using the P3P+MSAC algorithm (Section III-C). To illustrate the method description, we show pseudocode in Algorithm 1 that describes step by step the data processing in each stage.

A. SENSOR DATA PREPROCESSING

1) CAMERA IMAGE PROCESSING

For each camera image ($\mathbf{I}$), we must calculate the edge intensity value of each of its pixels. In outdoor environments, we often find areas with high lighting intensity, causing other areas of the same image to become very shaded, thus losing the edge definition. To improve the contrast of shaded areas, we apply histogram equalization to $\mathbf{I}$ in the first step of image preprocessing. In the *top-right* image in Fig. 3, we show the effect of applying this equalization and how the shaded areas become more defined. Due to both LiDAR and camera sensors can see objects from different points of view, the edges of these objects may not coincide exactly. For this reason, we apply a Gaussian filter to the equalized image to widen the edges, in other words, the filter helps to smoothing the objective function for maximizing edge alignment. With this filter, we do not improve alignment efficiency but we help the convergence of the algorithm (Fig. 3 *bottom-left*). Finally, we apply the gradient operator Sobel to a filtered image to obtain $\mathbf{E}$, which contains the edge intensity information of $\mathbf{I}$. To cover all possibilities in alignment with LiDAR discontinuities, we consider both horizontal and vertical edges in the Sobel operator application. We show in *bottom-right* image of Fig. 3, an example of $\mathbf{E}$ achieved from an image $\mathbf{I}$ in an arbitrary environment.

2) LiDAR POINT CLOUD PROCESSING

Assuming that the LiDAR point cloud $\mathbf{P}^L$ is layered, we base our filtering method on the one described in [7]. According to our filter definition expressed in (2), for each element $\mathbf{p}_i \in \mathbf{P}^L$, we calculate its corresponding $d_i \in \mathbf{d}$, that is, its discontinuity intensity associated.

$$d_i = max(r_{i-1} - r_i, r_{i+1} - r_i, 0) \tag{2}$$

where $r_i = |\mathbf{p}_i|$ is the range value, that is, the euclidean distance to i-th point, and r_{i-1} and r_{i+1} are the neighbours of r_i that are in the same scan layer. Then, we project this value of discontinuity intensity on the image plane described by the camera intrinsic parameters $\mathbf{K}$, and the initial guess, by using (1), obtaining, as a result, the image $\mathbf{D}$. This image contains a channel with the depth information of each $\mathbf{p}_i$ projected on the image plane, and a second channel with the corresponding d_i value calculated in (2). As an example, we show in the *top* image in Fig. 4 a point cloud $\mathbf{P}^L$ in an arbitrary environment. The corresponding projection in image $\mathbf{D}$ is shown in the *bottom* image in Fig. 4, in which the intensity of the green color component is proportional to the discontinuity intensity at the point.

B. CO-REGISTRATION OF CAMERA-LIDAR POINTS

To perform edge alignment at the object level, we do a clustering at $\mathbf{D}$ points, first in the depth dimension, and then, in two dimensions of the image plane. In this way, we can mitigate the effect of distortions produced by miscalibration on the LiDAR points projected. Some of these clusters represent clearly defined objects that provide us a lot of contours favorable for the co-registration. In contrast, other clusters present widely scattered points that provide poor information. To rule out these latter cases, we quantified the entropy in each cluster to exclude ones that present a high degree of disorder in his distribution. For entropy measurement, we used the formulation described in [43]. We show the clustering computation described, in line 16 of Algorithm 1, where $\mathbf{D}$ is the depth and discontinuities image, and $\mathbf{C} = (\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_m)$ are

Algorithm 1 Targetless Extrinsic Calibration Method

Inputs
$\mathbf{I}$:= Grayscale image.
$\mathbf{P}^L$:= Point cloud. {Where $\mathbf{P}^L = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$, and $\mathbf{p}_i$ is a 3D point.}
$\mathbf{\Theta}_0^{CL} \in \mathbb{R}^{6\times1}$:= Initial guess.
$\mathbf{K}$:= Intrinsic parameters.

Outputs
$\Delta\hat{\mathbf{\Theta}}^C \in \mathbb{R}^{6\times1}$:= Estimated calibration parameters.

Step A: Sensor data preprocessing
$\mathbf{E}$ = GetImageEdges($\mathbf{I}$);
$\mathbf{d}$ = GetPointDiscontinuities($\mathbf{P}^L$); {Where $\mathbf{d} = (d_1, d_2, \ldots, d_n)$, and d_i is a discontinuity intensity}
$\mathbf{D}$ = ProjectPonitsOnImagePlane($\mathbf{P}^L$, $\mathbf{d}$, $\mathbf{\Theta}_0^{CL}$, $\mathbf{K}$);

Step B: Co-registration of camera-Lidar points
$\mathbf{C}$ = ObjectClustering($\mathbf{D}$); {Where $\mathbf{C} = (\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_m)$}
$\mathbf{V}$ = SelectClustersReferenceVectors($\mathbf{C}$); {Where $\mathbf{V} = (\vec{\mathbf{v}}_1, \vec{\mathbf{v}}_2, \ldots, \vec{\mathbf{v}}_m)$}
$\mathbf{U}$ = SelectAllPosibleEdgesVectors($\mathbf{E}$); {Where $\mathbf{U} = (\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \ldots, \vec{\mathbf{u}}_k)$}
for i = 1: m **do**
  for j = 1: k **do**
    $\mathbf{C}_{ij}$ = ClusterPointsTransform($\mathbf{C}_i$, $\vec{\mathbf{v}}_i$, $\vec{\mathbf{u}}_j$));
    J_j = ObjectiveFunction($\mathbf{C}_{ij}$, $\mathbf{E}$);
  end for
  j = FindIndex($\mathbf{J}$ = Maximum($\mathbf{J}$))
  $\mathbf{C}_i^*$ = ClusterPointsTransform($\mathbf{C}_i$, $\vec{\mathbf{v}}_i$, $\vec{\mathbf{u}}_j$));
end for
$\mathbf{C}^*$:= Transformated clusters. {Where $\mathbf{C}^* = (\mathbf{C}_1^*, \mathbf{C}_2^*, \ldots, \mathbf{C}_m^*)$}

Step C: Relative transform estimation
$\mathbf{P}^C$ = 3DTo2DCorrespondence($\mathbf{C}^*$, $\mathbf{P}^L$, $\mathbf{\Theta}_0^{CL}$); {Where $\mathbf{P}^C = (\mathbf{P}_1^C, \mathbf{P}_2^C, \ldots, \mathbf{P}_m^C)$}
$\Delta\hat{\mathbf{\Theta}}^C$ = RelativeTransformEstimation($\mathbf{C}^*$, $\mathbf{P}^C$, $\mathbf{K}$);

FIGURE 3. Image preprocessing for data readiness for the co-registration stage. ***Top-left***: Original image taken in grayscale. ***Top-right***: Original image after applying histogram equalization. The central part that looked very dark in the original image now shows a higher contrast. ***Bottom-left***: Result of applying a Gaussian filter to the equalized image. Thanks to this process, the image edges are wider so that the alignment is not too restrictive. ***Bottom-right***: Final result of the process that provides the image **E**. For this process, we apply the Sobel gradient operator in order to obtain the edge intensity at each point of the image.

clusters calculated. Each $\mathbf{C}_i \in \mathbf{C}$ contains the location and the intensity discontinuity of each own point. The number of clusters m is variable and depends on the number of objects in the environment and their spacing. Fig. 5 *top* shows in yellow points several examples of clusters calculated in an arbitrary environment.

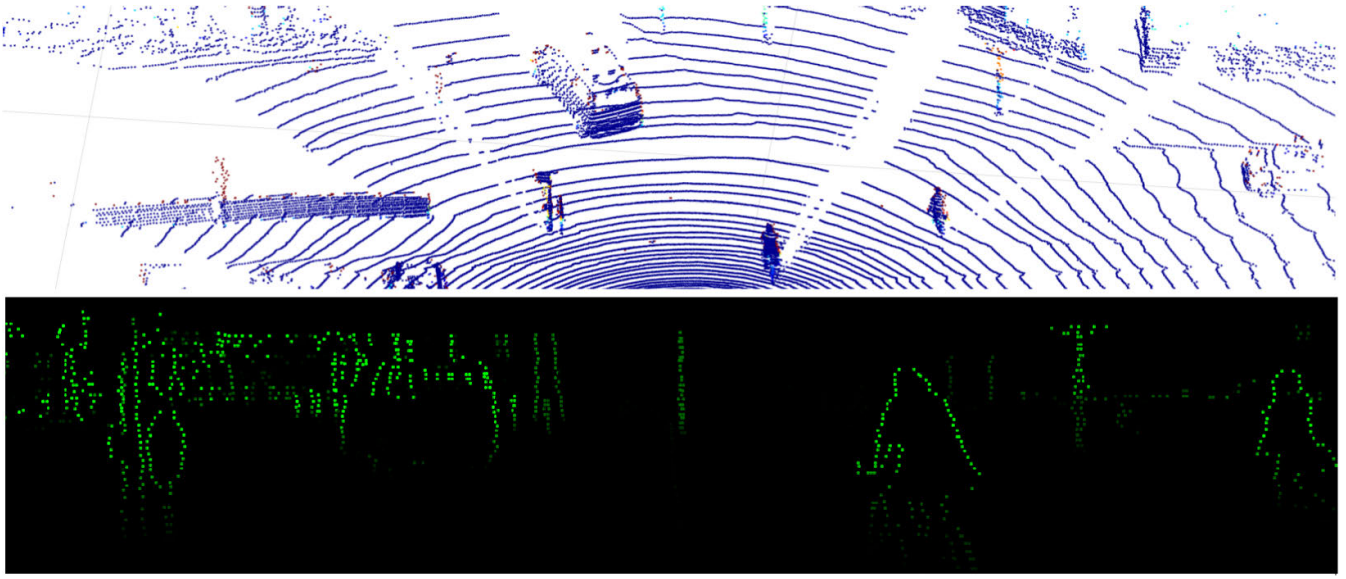

FIGURE 4. LiDAR point cloud preprocessing for data readiness for point co-registration. *Top*: 3D representation of an original point cloud captured with a LiDAR sensor in an arbitrary environment. *Bottom*: The same point cloud after applying the filtering described in (2). We project the discontinuity intensity values $\mathbf{d}$ with a proportional green color intensity on the image $\mathbf{D}$. We have made the projection of the points on the image plane using the initial guess.

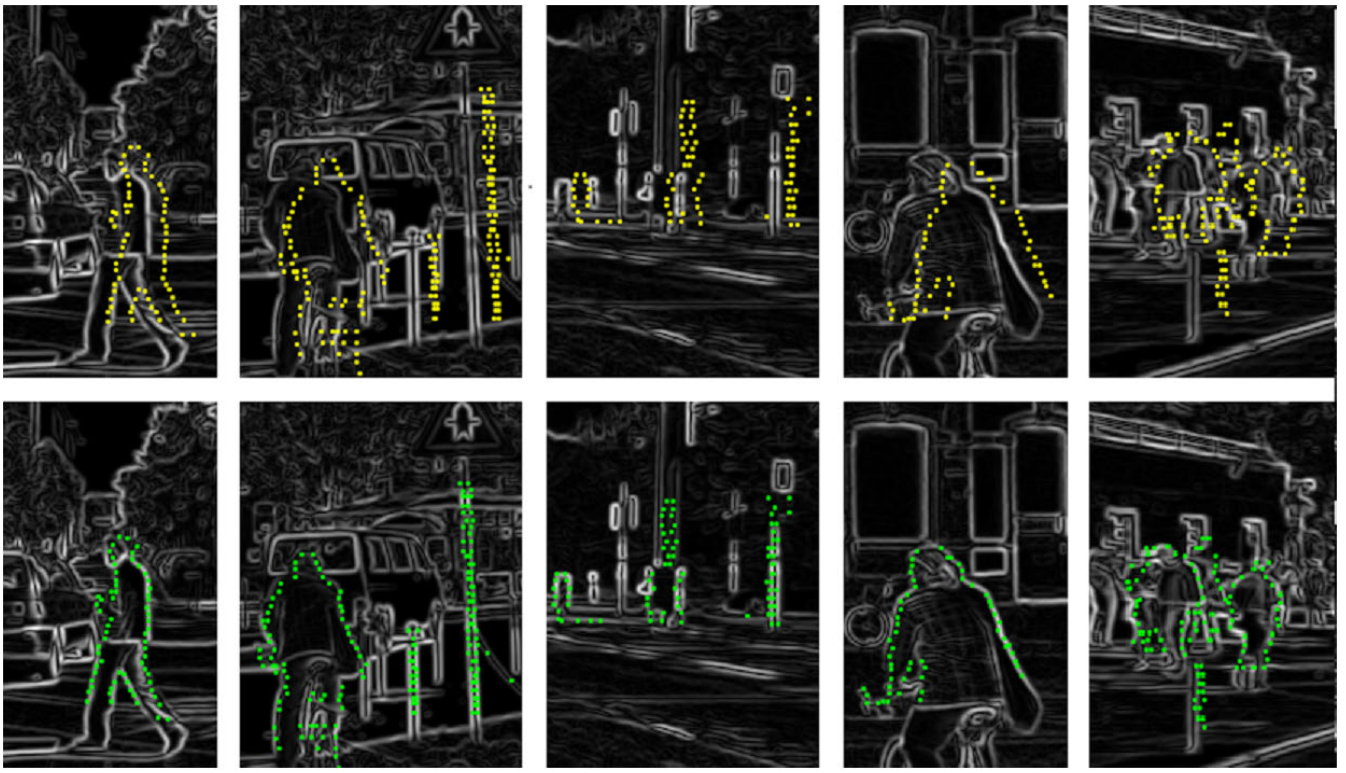

FIGURE 5. Example of a co-registration of the LiDAR points with points in the image. *Top*: In these five images, we show the points belonging to different clusters $\mathbf{C}_i$ projected in the image (yellow points), for a data sequence with some miscalibration. In this case, the same objects captured by both sensors do not coincide in the image plane. *Bottom*: In these five images, we show, in green color, the points belonging to the clusters $\mathbf{C}_i^*$ obtained from the best transformation calculated by maximizing (6). After applying this transformation, we can see the alignment of the objects in the image plane. In these cases, we can define the reprojection errors by the distance between each yellow point with its corresponding transformed green point.

For each $\mathbf{C}_i \in \mathbf{C}$ cluster, we select a pair of its own points, which is considered as a reference vector of these cluster: $\vec{\mathbf{v}}_i = (u_{i_2} - u_{i_1}, v_{i_2} - v_{i_1})$, where (u, v) are coordinates in image plane. The criteria for a pair of points selection is not quite relevant. Only a minimum and maximum range of separation between the pair of points have to be taken into account. The calculation of cluster reference vectors is computed in line 17 of Algorithm 1, where $\mathbf{V} = (\vec{\mathbf{v}}_1, \vec{\mathbf{v}}_2, \ldots, \vec{\mathbf{v}}_m)$ is a set

Algorithm 2 Reference Vector Calculation for $\mathbf{E}$

```
1: function SelectAllPosibleEdgesVectors(E)
2: ϵ := Edge intensity threshold (configurable parameter).
3: n := Number of pixels in E graters than ϵ.
4: k = 0;
5: for i = 1: n do
6:    for j = 1: n do
7:       (u1, v1) = GetPixelCoordinate(E(i));
8:       (u2, v2) = GetPixelCoordinate(E(j));
9:       ūk = (u2 − u1, v2 − v1);
10:      U = Append(U, ūk);
11:      k = k + 1;
12:   end for
13: end for
14: return U;
15: end function
```

of vectors. Moreover, we can describe $\mathbf{E}$ as a set of reference vectors $\mathbf{U} = (\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \ldots, \vec{\mathbf{u}}_k)$ arising from all possible k pairs of points that presents a certain edge intensity. In Algorithm 2, we show a pseudocode that illustrates the procedure to obtain these reference vectors. This pseudocode is computed in line 18 of Algorithm 1.

Given $\vec{\mathbf{v}}_i \in \mathbf{V}$ and $\vec{\mathbf{u}}_j \in \mathbf{U}$, it is possible to define the translation (3), rotation (4), and scaling (5) that transform, in the 2D image plane, **all the points** of the i-th cluster given by $\vec{\mathbf{v}}_i$ to the space defined by $\vec{\mathbf{u}}_j$.

$$\vec{\mathbf{t}}_{ij} = (u_{i_1} - u_{j_1}, v_{i_1} - v_{j_1}) \tag{3}$$

$$\alpha_{ij} = \arccos \frac{\vec{\mathbf{v}}_i \vec{\mathbf{u}}_j}{|\vec{\mathbf{v}}_i||\vec{\mathbf{u}}_j|} \tag{4}$$

$$w_{ij} = \frac{|\vec{\mathbf{u}}_j|}{|\vec{\mathbf{v}}_i|} \tag{5}$$

Then, to find the best alignment of a particular object in $\mathbf{C}_i$ with entire $\mathbf{E}$, we must find the best transformation from $\vec{\mathbf{v}}_i$ to all $\vec{\mathbf{u}}_j \in \mathbf{U}$, in terms of an objective function. In practice, a high value of k might cause execution times to be excessively large. However, since an initial guess is available, the choice of reference vectors can be relaxed for each cluster, performing the search in a limited range of α_{ij} and w_{ij}, as well as a convergence area defined by $\vec{\mathbf{t}}_{ij}$ limitations. In lines 19-26 of Algorithm 1, we show the pseudocode that illustrates the edge alignment procedure, where the objective function that must be maximized (see line 22) computes the following expression:

$$J = \sum_{l=1}^{n(c)} e_l c_l \tag{6}$$

where $c_l \in \mathbf{C}_{ij}$ are the discontinuity intensity values of the cluster, and $e_l \in \mathbf{E}$ are the edge intensity values in $\mathbf{E}$ in which the c_l points are projected. The output of this co-registration stage is a set of $\mathbf{C}_i^* \in \mathbf{C}^*$ that contains $\mathbf{C}_i$ points transformed to a location where we maximize the edge alignment. To illustrate this edge maximization, we show several examples of $\mathbf{C}_i$ clusters in the *top* image of Fig. 5, while the *bottom* image of the same figure shows the clusters $\mathbf{C}_i^*$ aligned with $\mathbf{E}$ after applying to them the transformation computed by the **Step B** shown in Algorithm 1.

C. 3D-TO-2D RELATIVE TRANSFORM ESTIMATION

We define reprojection errors as the distances in pixels between the points $\mathbf{C}_i$ and $\mathbf{C}_i^*$ (Fig. 5). Then, the estimation of the extrinsic parameters will be the one that minimizes these errors. In this work, we use the 3D-to-2D relative motion estimation, from its current position defined by $\mathbf{\Theta}_0^{CL}$ to the position that minimizes reprojection errors. For now on, we name this relative translation as $\Delta\mathbf{\Theta}^C$. We assume that, for each cluster $\mathbf{C}_i$, the correspondences of each point with $\mathbf{P}^L$ are available. Then, we can define the i-th clusters represented in the lidar space (L) as $\mathbf{P}_i^L$. We use the next transformation in order to express the points in the camera frame using the current calibration: $\mathbf{P}_i^C = [\mathbf{R}|\mathbf{t}]_0^{CL}\mathbf{P}_i^L$. Furthermore, the points $\mathbf{C}_i^*$ on the image plane area also available, and these define the locations where we 'desire' to project $\mathbf{P}_i^C$. To obtain a unique transform, we merge all clusters; then, henceforth, we remove the subscript i. In this way, making use of (1), it is possible to define the following expression:

$$\mathbf{C}^* = \mathbf{K}\Delta[\mathbf{R}|\mathbf{t}]^C\mathbf{P}^C \tag{7}$$

where $\Delta[\mathbf{R}|\mathbf{t}]^C = f(\Delta\mathbf{\Theta}^C)$. We know all the elements of the expression (7) in the absence of $\Delta[\mathbf{R}|\mathbf{t}]^C$, which represents the relative movement in the calibration space. To obtain the best solution of (7), we use the P3P [38] algorithm based on PnP, which requires a minimum of 3 inliers to perform the estimation. Also, we use a variant of RANSAC (MSAC [40]) to detect inliers in estimation. With all this information, we can obtain a handy quality measure to update the calibration filter that we will describe in the next section.

IV. CALIBRATION FILTERING

Once $\Delta[\mathbf{R}|\mathbf{t}]$ has been estimated, it is time for the correction stage of $[\mathbf{R}|\mathbf{t}]$, which is given by:

$$[\mathbf{R}|\mathbf{t}]_n = [\mathbf{R}|\mathbf{t}]_0\Delta[\mathbf{R}|\mathbf{t}]_n \tag{8}$$

where n subscript indicates the time sequence index in the complete calibration system. As we show in (8), in the case of an unfavorable estimation of $\Delta[\mathbf{R}|\mathbf{t}] = f(\Delta\mathbf{\Theta})$, the current calibration would suffer an unwanted deviation. This scenario will occur continuously in outdoor environments because the real world is noisy. For this reason, we apply a Kalman filter so that the final calibration shows stable behavior, and we only update this calibration to the extent of the quality of the results obtained from the method explained in Section III. Below we describe, first, our Kalman filter model and, second, the proposed methodology for noise observations characterization.

A. KALMAN FILTER MODEL

In this subsection, we describe the Kalman filter proposed to filter the final calibration state using the observations obtained through our method. Therefore, we consider, for now on, the output of the method described in Fig. 2 as

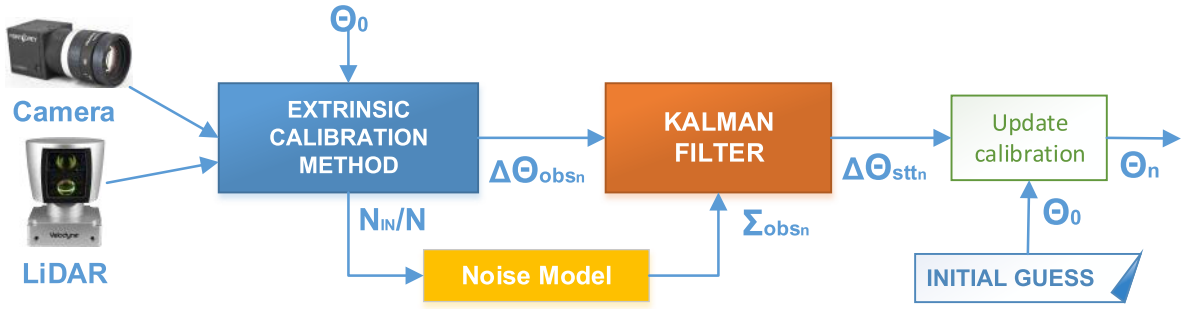

FIGURE 6. Complete calibration system. The blue block represents the method for obtaining extrinsic parameters, as shown in Fig. 2. We consider the output of this block as observation in a dynamic system that we filter with a Kalman filter (orange block). The yellow block represents the observation's noise model described in Section IV-B. The initial guess is the calibration with which the whole process starts, that is, $[\mathbf{R}|\mathbf{t}]_0 = f(\Theta_0)$. Expression (8) is applied in the update calibration block to obtain the current n-th calibration.

an observation in a dynamic system defined as: $\mathbf{Y} \sim N(\Delta\boldsymbol{\Theta}_{obs}, \Sigma_{obs})$. Then, we model the final calibration as 6-dimensional state defined as $\mathbf{X} \sim N(\Delta\boldsymbol{\Theta}_{stt}, \Sigma_{stt})$.

The Kalman filter proposed is based on a constant propagation model:

$$\Delta\hat{\boldsymbol{\Theta}}_{stt_n} = \Delta\boldsymbol{\Theta}_{stt_{n-1}} \tag{9}$$

This model does not have propagation additive noise. Then, the prediction stage of the Kalman filter is not computed, therefore $\Delta\boldsymbol{\Theta}_{stt}$ and Σ_{stt} will remain constant as long as we do not receive observation in the system. Although in the future, we could add an additive noise according to some criteria, such as perturbations caused by terrain, so that Σ_{stt} can grow over time if we don't receive observations.

We define the observation model as:

$$\Delta\hat{\boldsymbol{\Theta}}_{obs_n} = \Delta\hat{\boldsymbol{\Theta}}_{stt_n} + \delta_{obs_n} \tag{10}$$

where $\delta_{obs_n} \sim N(0, \Sigma_{obs_n})$ is the observation noise. We will describe the way to characterize this noise in Section IV-B, where we define the covariance matrix as a function of inlier detection in relative transform stage of our calibration method.

Given the proposed model (9)(10), we can compute the current state of the system as:

$$\Delta\boldsymbol{\Theta}_{stt_n} = f(\Delta\boldsymbol{\Theta}_{stt_{n-1}}, \Delta\boldsymbol{\Theta}_{obs_n}, \Sigma_{stt_{n-1}}, \Sigma_{obs_n}) \tag{11}$$

$$\Sigma_{stt_n} = f(\Sigma_{stt_{n-1}}, \Sigma_{obs_n}) \tag{12}$$

where (11) and (12) compute the innovation, Kalman gain, and state correction stages of the Kalman filter only each time in which we receive an observation.

At the beginning of the process, we don't have the confidence of the initial guess, i.e., we don't have a value of Σ_{stt_0} associated to $\Delta\boldsymbol{\Theta}_{stt_0}$. For this reason, we can initialize the filter with a high covariance Σ_{stt_0} so that it converges as we compute the observations $\mathbf{Y} \sim N(\Delta\boldsymbol{\Theta}_{obs}, \Sigma_{obs})$. Once we initiate the filter process, we should take into account that high values in state covariance Σ_{stt} are commonly associated with a wrong state, and useful observations are usually associated with low magnitudes in Σ_{obs}. In these cases, the filter relies more on observation than in the previous state. Then a set of useful observations along the time can attract step by step the estimations to a fine state. In opposite cases, we observe a different behavior, because when we reached a fine state, the state covariance Σ_{stt} usually takes low values, and the filter does not rely on lousy observation with high components in Σ_{obs}. This behaviour is what brings high stability to the system. We consider the system calibrated when we reach a state covariances below a particular configurable value. This threshold defines how precisely we think the system converges. But even if we consider the system calibrated, the filter should still work since we orient our method to online operation, where, as we have already mentioned, the system may suffer from disturbances that the calibration must correct. We define as transitory phases those parts of the process in which the system can find miscalibrated. In the results section, we quantify these transitory parts by measuring the number of data frames that the system needs to converge to a stationary phase. We show the block diagram of the complete calibration system in Fig. 6, which includes both the method described in Fig. 2 and the Kalman filter proposed in this section.

B. OBSERVATION's NOISE CHARACTERIZATION

As we mentioned in previous Section, we can express the results of our calibration method as observations in a dynamic system: $\mathbf{Y} \sim N(\Delta\boldsymbol{\Theta}_{obs}, \Sigma_{obs})$, where:

$$\Delta\boldsymbol{\Theta}_{obs} = (\Delta t_x, \Delta t_y, \Delta t_z, \Delta\theta_x, \Delta\theta_y, \Delta\theta_z) \tag{13}$$

$$\Sigma_{obs} = diag(\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_{\theta_x}^2, \sigma_{\theta_y}^2, \sigma_{\theta_z}^2) \tag{14}$$

where each σ^2 is the variance associated to each observation variable in $\Delta\boldsymbol{\Theta}_{obs}$. Due to the noise from the system affects with a different distribution, on the one hand, to translation, and, on the other hand, to the rotation, we can redefine (14) as:

$$\Sigma_{obs} = diag(\sigma_t^2, \sigma_t^2, \sigma_t^2, \sigma_\theta^2, \sigma_\theta^2, \sigma_\theta^2) \tag{15}$$

To characterize noise, we should calculate σ_t^2 and σ_θ^2 depending on the reliability of the calibration observation. As mentioned in the method description, we observe in our experiments that exist relationships between the percentage of inliers detected in the 3D-to-2D relative transform estimation and the accuracy of the calibration estimation.

Algorithm 3 Noise Filter Modeling

1: **function** NoiseFilterModeling($\mathcal{D}$, $\mathbf{\Theta}_{GT}$)
2: $\mathcal{D}$:= Camera-LiDAR data $\mathcal{D} = (D_1, D_2, \ldots, D_M)$.
3: $\mathbf{\Theta}_{GT}$:= Ground truth calibration.
4: $\mathbf{e}_t$:= Translation error vector $\mathbf{e}_t = (e_{t_1}, e_{t_2}, \ldots, e_{t_{100}})$.
5: $\mathbf{e}_\theta$:= Rotation error vector $\mathbf{e}_\theta = (e_{\theta_1}, e_{\theta_2}, \ldots, e_{\theta_{100}})$.
6: σ_t := Translation st. deviation $\sigma_t = (\sigma_{t_1}, \sigma_{t_2}, \ldots, \sigma_{t_{100}})$.
7: σ_θ := Rotation st. deviation $\sigma_\theta = (\sigma_{\theta_1}, \sigma_{\theta_2}, \ldots, \sigma_{\theta_{100}})$.
8: $\mathbf{c}$:= Counter vector $\mathbf{c} = (c_1, c_2, \ldots, c_{100})$.
9: $(\mathbf{e}_t, \mathbf{e}_\theta, \mathbf{c})$ = InitializeToZero $(\mathbf{e}_t, \mathbf{e}_\theta, \mathbf{c})$;
10: $(\sigma_t, \sigma_\theta)$ = InitializeToZero $(\sigma_t, \sigma_\theta)$;
11: **for** i = 1: M **do**
12: $(\Delta\mathbf{\Theta}, N, N_{IN})$ = CalibrationMethod$(D_i, \mathbf{\Theta}_{GT})$;
13: $\epsilon_t = |(\Delta t_x, \Delta t_y, \Delta t_z)|$;
14: $\epsilon_\theta = |(\Delta\theta_x, \Delta\theta_y, \Delta\theta_z)|$;
15: j = Round$((N_{IN}/N) \times 100)$;
16: $c_j = c_j + 1$;
17: $e_{t_j} = e_{t_j} + \epsilon_t$;
18: $e_{\theta_j} = e_{\theta_j} + \epsilon_\theta$;
19: **end for**
20: $\mathbf{e}_t = \mathbf{e}_t/\mathbf{c}$;
21: $\mathbf{e}_\theta = \mathbf{e}_\theta/\mathbf{c}$;
22: $(\sigma_t, \sigma_\theta)$ = FitExponentialFunction$(\mathbf{e}_t, \mathbf{e}_\theta)$;
23: **return** $\sigma_t^2, \sigma_\theta^2$;
24: **end function**

Then, we can consider the rate of inliers in co-registered points as reliability measurements. In Algorithm 3, we show how to model these relationships using a 'training' set of data $\mathcal{D}$, and a ground truth calibration $\mathbf{\Theta}_{GT}$. In the ideal case, if we compute our method (line 12 in Algorithm 3) using the ground truth calibration as an initial guess, we should obtain $\Delta\mathbf{\Theta} \triangleq 0$. Then, with this scenario, we can consider $\Delta\mathbf{\Theta}$ as a measure of the accuracy of our method, which can be express as a function of the percentage of inliers. In Algorithm 3, line 15, we compute the inliers rate, where N indicates the amount of 3D-to-2D co-registered points, and N_{IN} are the number of inliers in the relative transform estimation stage. We express the relationships between accuracy and inliers rate in Algorithm 3 as vectors $\mathbf{e}_t$ and $\mathbf{e}_\theta$. Then, we fit an exponential function to these data to cover all possibilities of the inliers rate. Finally, we sample this exponential function in σ_t^2 and σ_θ^2 vectors. These vectors are the output of Algorithm 3 and are the noise model that we use in the Kalman filter process. As we can see in the yellow block in Fig. 6, each time we receive an observation in the filtration process, we can compute inliers rate j, and then, by consulting $\sigma_{t_j}^2$ and $\sigma_{\theta_j}^2$ modelled, we can compute Σ_{obs}.

V. EXPERIMENTS AND RESULTS

In this section, we present the experimental results achieved through the application of the calibration method proposed in this work. First, in Section V-A, we define the process of noise modeling for the Kalman filter observations. Second, in Section V-B, we show the quantitative results obtained by applying the method to different kinds of environments derived from the Kitti dataset [44], using the calibration provided by this dataset as ground truth. Also, we compare our results with other state-of-the-art works in the Section V-C.

We achieve the results exposed in this section by accumulating the co-registered points in the image plane every 10 frames. This frame number is a configurable parameter, not a constrain of our method. Due to the accumulation is done in the image plane, we don't require any other process, and we only need to store the 3D-to-2D co-registered points for each 10 frames. From now on, we name these frame groupings as *samples*.

A. NOISE FILTER MODELING

To model the observation noise for the experiments, we use a 'training' set $\mathcal{D}$ that contains 560 *samples* of camera-LiDAR data from the Kitty dataset, and the calibration provided by the dataset as ground truth $\mathbf{\Theta}_{GT}$. Then, given $\mathcal{D}$ and $\mathbf{\Theta}_{GT}$, we can compute Algorithm 3 to fit the model as we described in the previous section. In Fig. 7, we show the results of noise modeling, where blue bars represent the values of $\mathbf{e}_t$ and $\mathbf{e}_\theta$ obtained, respectively. The red lines drawn in Fig. 7 graphics represent the sampled noise model for these experiments, defined by σ_t and σ_θ. In Fig. 7, we can see that the noise model covers all possible values of the inliers rate. In practice, it is not quite like that, because the observations that present an inliers percentage below 5% are first consider invalid and then rejected.

B. METHOD VALIDATION

To test our method and its corresponding filtration system, we use the Kitti dataset, in which a ground truth calibration $\mathbf{\Theta}_{GT}$ is available. Then, using this calibration as initial guess, we can define the computation of our system as:

$$[\mathbf{R}|\mathbf{t}]_n = [\mathbf{R}|\mathbf{t}]_{GT}\,\Delta[\mathbf{R}|\mathbf{t}]_{stt_n} \tag{16}$$

where $\Delta[\mathbf{R}|\mathbf{t}]_{stt_0} = f(\Delta\mathbf{\Theta}_{stt_0})$ is a normal distributed random misscalibration. We inicialize the covariance matrix of the state as $\Sigma_{stt_0} = f(\sigma_{t_0}^2, \sigma_{\theta_0}^2)$, where $\sigma_{t_0} = 0.5m$ and $\sigma_{\theta_0} = 0.8°$ are the standard deviations with which we have generated the random misscalibration. Given this scenario, if we compute the system shown in Fig. 6, when n increases, $\Delta\mathbf{\Theta}_{stt_n}$ should tend to 0. In this way, we can evaluate our complete system as far as the state converges to 0 for all state variables. For this experiment, we select 300 *samples* from the Kitti dataset that contains data from four different types of environments, such as follows: campus, road, city, and residential. We divide the evaluation test into 3 subsets of 100 samples each one, and each subgroup contains data from the different types of environments. For each set, we compute the whole process shown in Fig. 6, initializing the process by causing a random miscalibration, as we mentioned at the beginning of this Section.

In Fig. 8 we show the value of the observations obtained from each sample of the subset 1. These observations are the

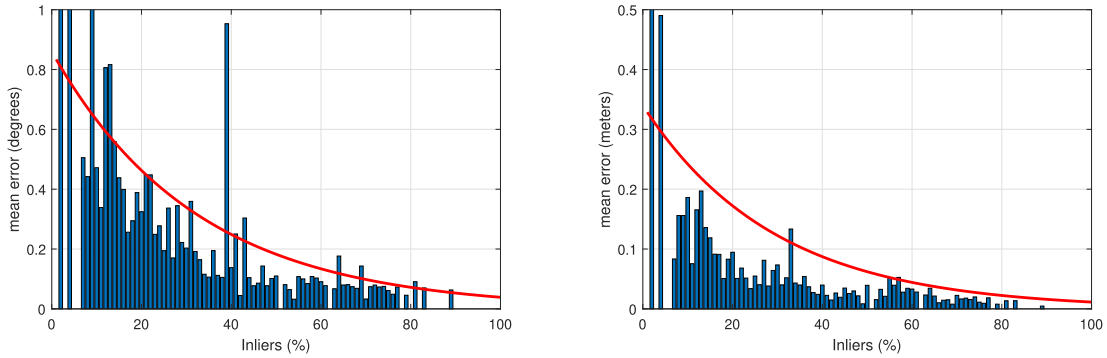

FIGURE 7. Noise model for the Kalman filter observations described in Section V-A. *Left*: noise model for the module of the three orientation components $\mathbf{r} = |(\theta_x, \theta_y, \theta_z)| = |(yaw, pitch, roll)|$. *Right*: noise model for the module of the three translation components $\mathbf{t} = |(x, y, x)|$. These graphs have been obtained by processing 5, 600 frames, by calculating the inlier percentage and the error with respect to the ground truth of each 10 frames processed. The blue bars indicate the mean error for each inlier percentage measured. The red lines describe the exponential function that fits the data.

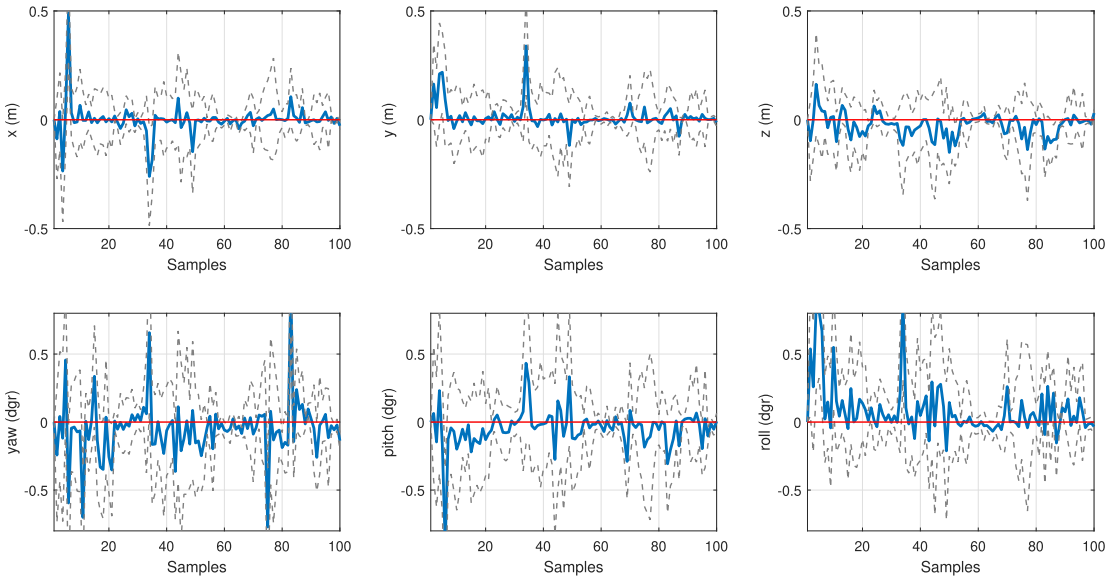

FIGURE 8. Representation of each variables that make up the observations $\Delta\Theta_{obs_n}$ for $n = [0, \ldots, 100]$. The blue line indicates the value that each variable takes at the instant n, the dashed lines mark the standard deviations associated with each observation, and the red line indicates the value of the desired state, which is 0 in all the variables in this case. Due to the noisy outdoor environments considered, the calibrations suffer a number of disturbances that must be filtered.

output of the method described in Section III. In this figure, the dashed line represents the standard deviation obtained in each measurement from the noise model described in Section V-A, while the red line represents the desired state. We can see that the method provides some calibrations translation close to 0, and, in these cases, the calibrations will be useful. However, these observations would generate little stable calibrations and with very abrupt variations along the time. In contrast, as we show in Fig. 9, if we apply the observations to the Kalman filter described in Section IV, the state presents smooth evolution, and reduce its noise so that more noisy observations will not cause miscalibration. We can see in Fig. 9 how the filter transforms the noisy information provided by the observations as a proper final calibration. In Fig. 9, the dashed lines indicate the standard deviation, and the red line indicates the desired state. In this case, we can see that the evolution of the calibration presents an initial transitory evolution until it converges to a particular calibration, and, from that point, it offers an evolution of a stationary nature.

We present the quantitative results for the translation and rotation variables, respectively, in Tables 1 and 2. We achieve these results from the evaluation of the complete method shown in Fig 6, for each of the 3 data subsets. To test the method behavior in terms of convergence, we measure the number of samples in the transitory part of the process. Then, we evaluate the mean of the error between the stationary part of the state evolution against the ground truth for each state variable. Also, we consider the standard deviation of the error calculated above. This last calculation provides information on the stability of the method in the stationary part of its evolution.

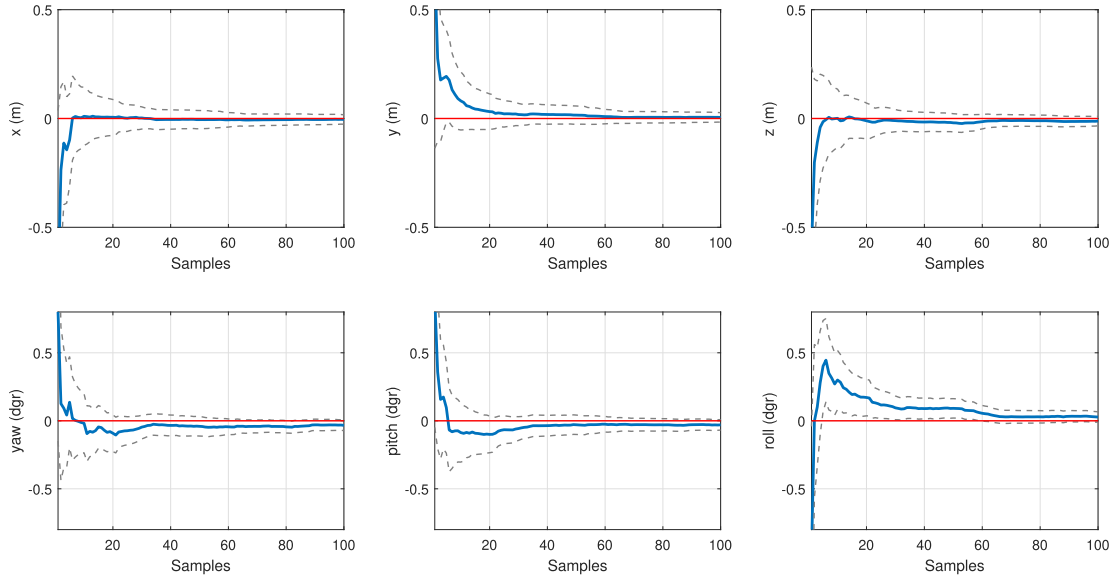

FIGURE 9. Representation of the time evolution of the state variables that make up $\Delta\Theta_{stt_n}$ for $n = [0, \ldots, 100]$. The blue line indicates the value that each variable takes at time n, the dashed lines indicate the standard deviations associated with each state, and the red line indicates the value of desired state, which in this case is 0 in all variables. This evolution arises from the application of the Kalman filter described in Section IV, sequentially for each observation shown in Fig 8. The process presents a transitory phase and a stationary phase in all cases. Furthermore, the system converges in all cases with very low errors and high stability against noise.

TABLE 1. Quantitative results for translation for three different data subsets. It can be seen how the number of transitory iterations is usually low, except in some exceptions such as the subset 1. The errors obtained are very low, and the standard deviation values indicate that the system has good stability.

Subset	State variables	Trans. interval ($samples$)	Estat. mean error (m)	Estat. std error (m)
1	x	35	-0.52×10^{-2}	0.08×10^{-2}
	y	63	0.70×10^{-2}	0.30×10^{-2}
	z	64	1.31×10^{-2}	0.39×10^{-2}
2	x	21	-0.63×10^{-2}	0.13×10^{-2}
	y	18	0.09×10^{-2}	0.06×10^{-2}
	z	31	-0.96×10^{-2}	0.29×10^{-2}
3	x	30	-0.52×10^{-2}	0.09×10^{-2}
	y	22	0.12×10^{-2}	0.09×10^{-2}
	z	23	-0.43×10^{-2}	0.20×10^{-2}

TABLE 2. Quantitative results for rotation for three different data subsets. In this case, convergence is somewhat slower than in the case of translation. However, as in the case of translation, the errors are very low, and the standard deviation values indicate that the system also has good stability against rotation noise.

Subset	State variables	Trans. interval ($samples$)	Estat. mean error ($degrees$)	Estat. std error ($degrees$)
1	yaw	45	-3.96×10^{-2}	0.57×10^{-2}
	$pitch$	40	-3.00×10^{-2}	0.19×10^{-2}
	$roll$	64	4.15×10^{-2}	1.98×10^{-2}
2	yaw	53	-3.65×10^{-2}	0.41×10^{-2}
	$pitch$	41	-2.38×10^{-2}	0.24×10^{-2}
	$roll$	30	0.12×10^{-2}	0.47×10^{-2}
3	yaw	26	-2.73×10^{-2}	0.47×10^{-2}
	$pitch$	31	-3.18×10^{-2}	0.12×10^{-2}
	$roll$	38	2.51×10^{-2}	0.48×10^{-2}

We can see that the number of transitory iterations is quite variable in the results presented in Tables 1 and 2. This variability is due to the fact that the transitory length depends mainly on the noise in the first stack of observations. In other words, the system will remain in a transitory regime until we receive a set of low-noise observations. In some cases, such as the variables y and *roll* of the subset 1, we can see that the transitory phase is much larger than in other cases. Nevertheless, if we see the evolution of these variables in Fig. 9, previously the system has reached a semi-stable state that we could consider as a *pre-stationary* regime. We can also observe in both cases how the system converges when we receive low-noise observations. We can consider the average errors that we achieve as shallow, most of them bellow $1cm$ in translation and bellow $0.05°$ in rotation. The behavior presents a high degree of stability, which indicates that the system does not suffer considerable disturbances in the face of very noisy observations when it has reached the stationary regime.

C. METHOD COMPARISON

In the previous section, we have validated the behavior of our method. In addition, in order to test how our work contributes to the calibration field, we compare our approach with other state-of-the-art ones, that we consider seminal works. In Table 3, we show these works and a comparison in terms of the implementation strategy. We should note that, in the right column of Table 3, we show an indicator of the execution mode for each work. We consider that the method presents an execution mode capable of operating online when the algorithm converges along different sample-times, what allows us to track smooth deviations in calibration. As an online example, the method proposed by *Levinson & Thrun* [7] computes only one gradient step each sample-time. Then, the edge maximization converges asymptotically. In contrast,

TABLE 3. Comparison of our method with other state-of-the-art ones, in terms of the implementation strategy. We can distinguish, as the most differentiating aspect, between methods that work in online mode, and those that do not. Our method is the only one that works with local features.

METHOD	IMAGE FEATURE	LiDAR FEATURE	OBJECTIVE FUNCTION	ONLINE MODE
Levinson & Thrun [7]	Global edges	Global Discontinuities	Edge alignment	Yes
Pandey [33]	Global luminance	Global reflectance	Mutual Information	No
Kang & Doh [32]	Global edges	Global Discontinuities	Edge alignment + GMM	No
Proposed method	Local edges	Local discontinuities	3D-to-2D co-register errors	Yes

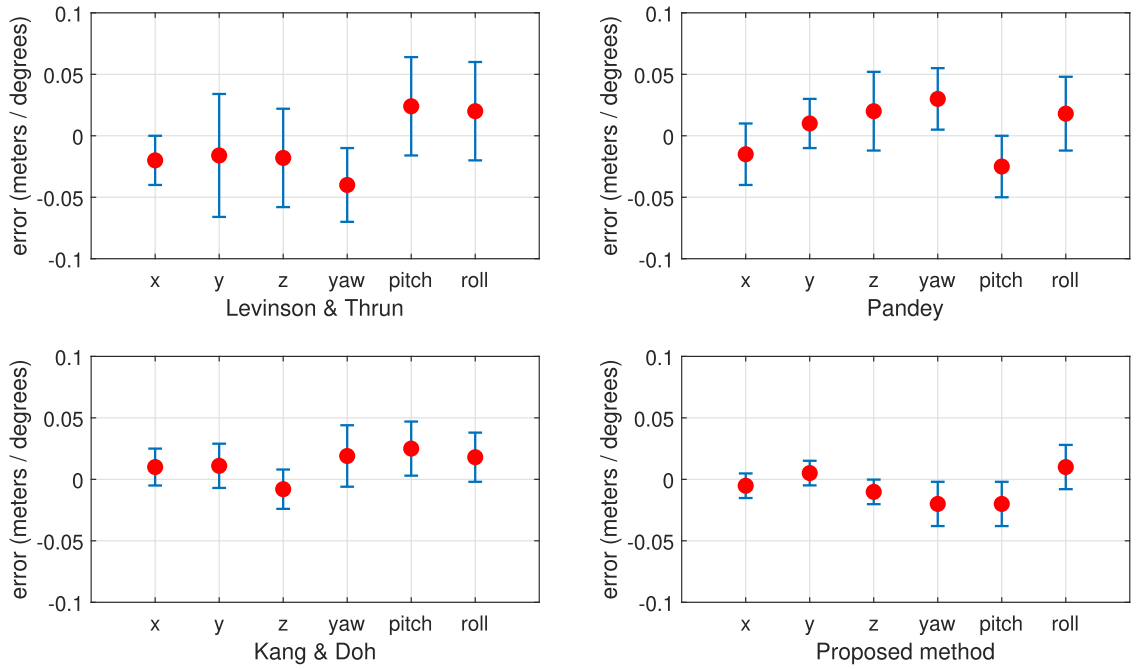

FIGURE 10. Experimental results for the comparison of our method with those proposed in Table 3. We can see that our method significantly improves the results with respect to those proposed by *Levinson & Thrun* [7] and *Pandey* [33]. In contrast, the results of *Kang & Doh* [32] are very close to ours, although with our method we achieved more precision in standard deviation. Our method also has the advantage of being online, and it can be used in autonomous navigation applications.

the methods proposed by *Pandey* [33] and *Kang & Doh* [32] need a set of samples and then compute the gradient descent iteratively, at the same sample-time, until the algorithm converges. We evaluated the different works using the Kitti dataset in different environments. As in the previous section, we have caused a (normal) random perturbation to a provided calibration, then we have applied the different methods to recalibrate the system. By using the ground truth, we have able to measure errors and their standard deviations. For these calculations in online methods, we computed 60 samples, 1 sample each time. And, for offline ones, we calculated 30 accumulated samples in a single calibration process.

In Fig. 10, we show the results obtained for each method. If we compare our method with the other online approach, we can see that we achieve a significant improvement in results. This improvement tells us that the local features strategy can benefit the system, as it can remove noisy parts of the scene that provide poor information to the process. In addition, we see that the results for the offline algorithms improve those of *Levinson & Thrun* [7], especially in the case of *Kang & Doh* [32]. Though its results are close to ours, we get more accurate standard deviation values. Moreover, although offline methods manage to converge using 30 samples, we consider the online use of our method a great advantage, since once the transitory phase is over, we need to compute only 1 sample every time-step to keep the system calibrated. Instead, with the offline methods, we would have to repeat the process with another 30 samples, which makes these methods very difficult to implement for real-time applications in mobile robotics.

VI. CONCLUSION AND FUTURE WORKS

In this article, we have presented a new method for automatic extrinsic calibration between camera and LiDAR sensors. We based our proposed approach on edge feature camera-LiDAR co-registration in arbitrary environments. Our method provides a first estimation of the parameters by minimizing 3D-to-2D errors obtained in the co-registration stage. Also, we have implemented a Kalman filter to smooth the evolution of the final calibration. By using the proposed complete calibration system, we have achieved shallow calibration errors against the ground truth provided by the state-of-the-art Kitti dataset. Furthermore, we have shown that the method is robust to observations with a high degree of noise, which provides an excellent stability for its online implementation in autonomous navigation applications.

Finally, we have compared our method with other state-of-the-art works, thus demonstrating the contributions of our work to the field of calibration.

As future work, we will address the implementation of the calibration system in a real autonomous vehicle. Specifically, in the BLUE vehicle owned by the AUROVA group to which we belong. This research platform is equipped with a Velodyne VLP16 LiDAR sensor and with a camera Realsense D-435. Both sensors could be calibrated online with the proposed method, also, taking into account potentials synchronization problems in sensor fusion. Furthermore, we will consider formalizing a noise model for state propagation in the Kalman filter, since the level of uncertainty increases as a function of the modeled noise in cases where there are no observations for a particular time sample. Our ultimate goal is to apply sensor fusion between the camera and LiDAR for scene understanding by using unsupervised clustering techniques.

REFERENCES

[1] H. Zhu, K.-V. Yuen, L. Mihaylova, and H. Leung, "Overview of environment perception for intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 10, pp. 2584–2601, Oct. 2017.

[2] Q. Deng, X. Li, P. Ni, H. Li, and Z. Zheng, "Enet-CRF-LiDAR: LiDAR and camera fusion for multi-scale object recognition," *IEEE Access*, vol. 7, pp. 174335–174344, 2019.

[3] M. Dimitrievski, P. Veelaert, and W. Philips, "Behavioral pedestrian tracking using a camera and LiDAR sensors on a moving vehicle," *Sensors*, vol. 19, no. 2, p. 391, Jan. 2019.

[4] Y.-S. Shin, Y. S. Park, and A. Kim, "Direct visual SLAM using sparse depth for camera-LiDAR system," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–8.

[5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[6] E.-S. Kim and S.-Y. Park, "Extrinsic calibration between camera and LiDAR sensors by matching multiple 3D planes," *Sensors*, vol. 20, no. 1, p. 52, Dec. 2019.

[7] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers," *Robot., Sci. Syst.*, vol. 2, p. 7, Jun. 2013.

[8] M. Á. Muñoz–Bañón, I. del Pino, F. A. Candelas, and F. Torres, "Framework for fast experimental testing of autonomous navigation algorithms," *Appl. Sci.*, vol. 9, no. 10, p. 1997, May 2019.

[9] I. del Pino, M. A. Mu noz-Ba non, S. Cova-Rocamora, M. Á. Contreras, F. A. Candelas, and F. Torres, "Deeper in BLUE," *J. Intell. Robotic Syst.*, vol. 98, no. 1, pp. 207–225, 2020.

[10] A. Geiger, F. Moosmann, O. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 3936–3943.

[11] E.-S. Kim and S.-Y. Park, "Extrinsic calibration of a camera-LIDAR multi sensor system using a planar chessboard," in *Proc. 11th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2019, pp. 89–91.

[12] R. Opromolla, M. Z. Di Fraia, G. Fasano, G. Rufino, and M. Grassi, "Laboratory test of pose determination algorithms for uncooperative spacecraft," in *Proc. IEEE Int. Workshop Metrology for Aerosp. (MetroAeroSpace)*, Jun. 2017, pp. 169–174.

[13] W. Wang, K. Sakurada, and N. Kawaguchi, "Reflectance intensity assisted automatic and accurate extrinsic calibration of 3D LiDAR and panoramic camera using a printed chessboard," *Remote Sens.*, vol. 9, no. 8, p. 851, Aug. 2017.

[14] 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 *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 3906–3912.

[15] L. Zhou, Z. Li, and M. Kaess, "Automatic extrinsic calibration of a camera and a 3D LiDAR using line and plane correspondences," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 5562–5569.

[16] J. Kummerle, T. Kuhner, and M. Lauer, "Automatic calibration of multiple cameras and depth sensors with a spherical target," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–8.

[17] J.-E. Ha, "Extrinsic calibration of a camera and laser range finder using a new calibration structure of a plane with a triangular hole," *Int. J. Control, Autom. Syst.*, vol. 10, no. 6, pp. 1240–1244, Dec. 2012.

[18] 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*, vol. 14, no. 3, pp. 5333–5353, Mar. 2014.

[19] X. Gong, Y. Lin, and J. Liu, "3D LIDAR-camera extrinsic calibration using an arbitrary trihedron," *Sensors*, vol. 13, no. 2, pp. 1902–1918, Feb. 2013.

[20] F. M. Mirzaei, D. G. Kottas, and S. I. Roumeliotis, "Analytical least-squares solution for 3d LIDAR-camera calibration," in *Robotics Research*. Cham, Switzerland: Springer, 2017, pp. 183–200.

[21] 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, 2020.

[22] S. Li, C. Xu, and M. Xie, "A robust O(n) solution to the perspective-n-point problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1444–1450, Jul. 2012.

[23] K. Strobl and G. Hirzinger, "Optimal hand-eye calibration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 4647–4653.

[24] K. Huang and C. Stachniss, "Extrinsic multi-sensor calibration for mobile robots using the Gauss-Helmert model," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 1490–1496.

[25] Z. Taylor and J. Nieto, "Motion-based calibration of multimodal sensor Extrinsics and timing offset estimation," *IEEE Trans. Robot.*, vol. 32, no. 5, pp. 1215–1229, Oct. 2016.

[26] Y. C. Shiu and S. Ahmad, "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX=XB," *IEEE Trans. Robot. Autom.*, vol. 5, no. 1, pp. 16–29, Feb. 1989.

[27] R. Ishikawa, T. Oishi, and K. Ikeuchi, "LiDAR and camera calibration using motions estimated by sensor fusion odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 7342–7349.

[28] C. Shi, K. Huang, Q. Yu, J. Xiao, H. Lu, and C. Xie, "Extrinsic calibration and odometry for camera-LiDAR systems," *IEEE Access*, vol. 7, pp. 120106–120116, 2019.

[29] J. Castorena, U. S. Kamilov, and P. T. Boufounos, "Autocalibration of LiDAR and optical cameras via edge alignment," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 2862–2866.

[30] K. Banerjee, D. Notz, J. Windelen, S. Gavarraju, and M. He, "Online camera LiDAR fusion and object detection on hybrid data for autonomous driving," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1632–1638.

[31] J. Jiang, P. Xue, S. Chen, Z. Liu, X. Zhang, and N. Zheng, "Line feature based extrinsic calibration of LiDAR and camera," in *Proc. IEEE Int. Conf. Veh. Electron. Saf. (ICVES)*, Sep. 2018, pp. 1–6.

[32] J. Kang and N. L. Doh, "Automatic targetless camera–LIDAR calibration by aligning edge with Gaussian mixture model," *J. Field Robot.*, vol. 37, no. 1, pp. 158–179, Jan. 2020.

[33] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *J. Field Robot.*, vol. 32, no. 5, pp. 696–722, Aug. 2015.

[34] K. Irie, M. Sugiyama, and M. Tomono, "Target-less camera-LiDAR extrinsic calibration using a bagged dependence estimator," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2016, pp. 1340–1347.

[35] J. Jeong, Y. Cho, and A. Kim, "The road is enough! Extrinsic calibration of non-overlapping stereo camera and LiDAR using road information," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2831–2838, Jul. 2019.

[36] T. Cui, S. Ji, J. Shan, J. Gong, and K. Liu, "Line-based registration of panoramic images and LiDAR point clouds for mobile mapping," *Sensors*, vol. 17, no. 12, p. 70, Dec. 2016.

[37] Q. Liao and M. Liu, "Extrinsic calibration of 3D range finder and camera without auxiliary object or human intervention," in *Proc. IEEE Int. Conf. Real-Time Comput. Robot. (RCAR)*, Aug. 2019, pp. 42–47.

[38] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 930–943, Aug. 2003.

[39] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *Proc. CVPR*, Jun. 2011, pp. 2969–2976.

[40] P. H. S. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Comput. Vis. Image Understand.*, vol. 78, no. 1, pp. 138–156, Apr. 2000.
[41] R. J. Meinhold and N. D. Singpurwalla, "Understanding the Kalman filter," *Amer. Statistician*, vol. 37, no. 2, pp. 123–127, 1983.
[42] F. Du and M. Brady, "Self-calibration of the intrinsic parameters of cameras for active vision systems," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 1993, pp. 477–482.
[43] X. Zhan, Y. Cai, and P. He, "A three-dimensional point cloud registration based on entropy and particle swarm optimization," *Adv. Mech. Eng.*, vol. 10, no. 12, 2018, Art. no. 1687814018814330.
[44] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

MIGUEL ÁNGEL MUÑOZ-BAÑÓN received the degree in telecommunications engineering from the University of Alicante, in 2016, and the M.S. degree in artificial intelligence with UNED, in 2017. He is currently pursuing the Ph.D. degree with the University of Alicante. He is currently a Research Technician with the University of Alicante. His research interests include machine learning, in particular, clustering through unsupervised learning applied to mobile robotics, and, especially, to graph-based simultaneous localization, and mapping (Graph-SLAM).

FRANCISCO A. CANDELAS received the Computer Science Engineering and the Ph.D. degrees from the University of Alicante (UA), in 1996 and 2001, respectively. He has been an Associate Professor with the Department of Physics, Systems Engineering, and Signal Theory, UA, since 1999, where he also researches with the Automatics, Robotics, and Computer Vision Group, since 1996. His research interests include autonomous robots, automation, and robotics education.

FERNANDO TORRES (Senior Member, IEEE) currently directs the Research Group Automatics, Robotics, and Computer Vision founded in 1996 at the University of Alicante. He is a member of TC 5.1 and TC 9.4 of the IFAC, and a CEA. Since 2018, he has been a Coordinator of electrical, electronic, and automatic evaluation with the ANECA-movility. His research interests include automation, robotics, and e-learning.