

# Improvements to Target-Based 3D LiDAR to Camera Calibration

Jiunn-Kai Huang and Jessy W. Grizzle

**Abstract**—The rigid-body transformation between a LiDAR and monocular camera is required for sensor fusion tasks, such as SLAM. While determining such a transformation is not considered glamorous in any sense of the word, it is nonetheless crucial for many modern autonomous systems. Indeed, an error of a few degrees in rotation or a few percent in translation can lead to 20 cm reprojection errors at a distance of 5 m when overlaying a LiDAR image on a camera image. The biggest impediments to determining the transformation accurately are the relative sparsity of LiDAR point clouds and systematic errors in their distance measurements. This paper proposes (1) the use of targets of known dimension and geometry to ameliorate target pose estimation in face of the quantization and systematic errors inherent in a LiDAR image of a target, (2) a fitting method for the LiDAR to monocular camera transformation that avoids the tedious task of target edge extraction from the point cloud, and (3) a “cross-validation study” based on projection of the 3D LiDAR target vertices to the corresponding corners in the camera image. The end result is a 50% reduction in projection error and a 70% reduction in its variance.

## I. INTRODUCTION AND RELATED WORK

The desire to produce 3D-semantic maps [1] with our Cassie-series bipedal robot [2] has motivated us to fuse 3D-LiDAR and RGB-D monocular camera data for autonomous navigation [3]. Indeed, by mapping spatial LiDAR points onto a segmented and labeled camera image, one can associate the label of a pixel (or a region about it) to the LiDAR point as shown in Fig. 1. An error of a few degrees in rotation or a few percent in translation in the estimated rigid-body transformation between LiDAR and camera can lead to 20 cm reprojection errors at a distance of 5 m when overlaying a LiDAR point cloud on a camera image. Such errors will lead to navigation errors.

In this paper, we assume that the intrinsic calibration of the two sensors has already been done [4] and focus on obtaining the rigid-body transformation, i.e. rotation matrix and translation, between a LiDAR and camera. This is a well studied problem with a rich literature that can be roughly divided into methods that require targets: [5], [6], [7], [8], [9], [10], [11] and those that do not: [12], [13], [14], [15], [16]. Our modest contributions are applicable to methods that use planar polygonal targets, such as checkerboards, triangles, and diamonds.

In target-based methods, one seeks to estimate a set of target features (e.g., edge lines, normal vectors, vertices) in the LiDAR’s point cloud and the camera’s image plane.

Jiunn-Kai Huang and J. Grizzle, are with the Robotics Institute, University of Michigan, Ann Arbor, MI 48109, USA. {bjhuang, grizzle}@umich.edu.



Fig. 1: Building a semantic map for autonomous navigation. Once the transformation from LiDAR to camera is known, it is possible to fuse the LiDAR and RGB-camera information to build a 3D-map that is annotated with semantic information. The lower right shows a single camera image; its segmentation with semantic labels is shown in the upper right. Synchronized LiDAR points are mapped onto the camera image, associated to a semantic label, and then re-projected to a local frame [20] along with the semantic labels to create a 3D semantically-labeled map [1], shown in the upper left. The sidewalk is marked in white, the grass in yellow-green, and trees in dark green. In the lower left, Cassie is using the map to plan a path around the North Campus Wave Field, while staying on the sidewalk.

If “enough” independent correspondences can be made, the LiDAR to camera transformation can be found by Perspective-n-Point (PnP) as in [17], that is, through an optimization problem of the form

$$(R_C^{L*}, T_C^{L*}) = \arg \min_{(R,T)} \text{dist}(P(H_L^C(X_i)), Y_i), \quad (1)$$

where  $X_i$  are the (homogeneous) coordinates of the LiDAR features,  $Y_i$  are the coordinates of the camera features,  $P$  is the often-called “projection map”,  $H_L^C$  is the (homogeneous representation of) the LiDAR-frame to camera-frame transformation with rotation matrix  $R_C^L$  and translation  $T_C^L$ , and  $\text{dist}$  is a distance or error measure.

## A. Rough Overview of the Most Common Target-based Approaches

The works closest to ours are [18], [19]. Each of these works has noted that rotating a square target so that it presents itself as a diamond can help to remove pose ambiguity due to the spacing of the ring lines; in particular, see Fig. 2 in [18] and Fig. 1 in [19]. More generally, we recommend the literature overview in [18] for a recent, succinct survey of LiDAR to camera calibration.

The two most common sets of features in the area of target-based calibration are (a) the 3D-coordinates of the vertices of a rectangular or triangular planar target, and (b) the normal vector to the plane of the target and the lines connecting the vertices in the plane of the target. Mathematically, these two sets of data are equivalent: knowing one of them allows the determination of the other. In practice, focusing on (b) leads

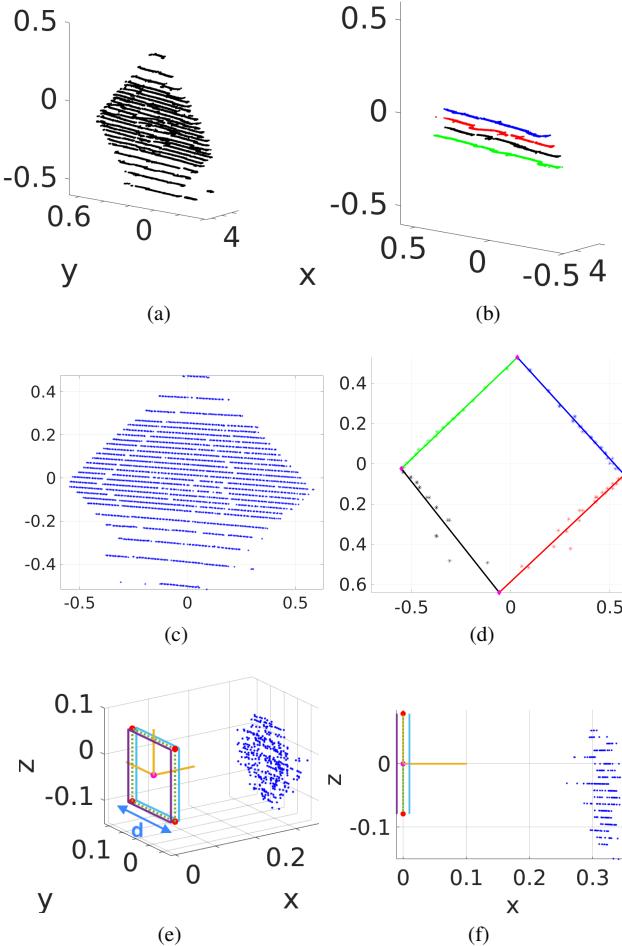


Fig. 2: (a) Twenty five LiDAR scans of a planar target. The point cloud is roughly 7 cm thick. (b) The “noise” in the point cloud is not random. A zoom for four of the rings (13, 15, 17, 19) is typical and shows systematic errors in distance. (c) LiDAR points orthogonally projected to the plane defined by the normal. (d) Example edge points selected to regress a line via *ransac*. (e) Target reference frame and real point cloud data. The dotted blue square is the reference target; its vertices are denoted  $\{\bar{X}_i\}_{i=1}^4$ . (f) A side-( $x - z$ )-view highlighting the non-zero thickness of a typical point cloud. These figures and all others in the paper are of sufficient resolution that they can be blown up to see detail.

to use of the SVD to find the normal vector and more broadly to least squares line fitting problems [19], while (a) opens up other perspectives, as highlighted in [18].

Figure 2a shows a 3D view of 25 scans from a factory-calibrated 32-Beam Velodyne ULTRA Puck LiDAR on a diamond shaped planar target, with a zoom provided in Fig. 2b. There are clearly systematic errors in the distance (or depth) measurement of the target, and these errors affect the estimation of the target’s “centroid”, which is commonly used to determine the translation of the target from the LiDAR and the target’s “normal vector”, which is used to define the plane of the target as in Fig. 2c. Subsequently, the point cloud is orthogonally projected to this plane and the line boundaries of the target are found by performing *ransac* on the appropriate set of ring edges; see Fig. 2d. The lines along the target’s

boundaries then define its vertices in the plane, which for later reference, we note are not constrained to be compatible with the target’s geometry.

Once the vertices in the plane of the target have been determined, then knowledge of the target’s normal allows the vertices to be lifted back to the coordinates of the point cloud. This process may be repeated for multiple scans of a target, aggregates of multiple scans of a target, or several targets, leading to a list of target vertices  $\{X_i\}_{i=1}^{4n}$ , where  $n$  is the number of target poses.

The target is typically designed so that the vertices are easily distinguishable in the camera image. Denoting their corresponding coordinates in the image plane by  $\{Y_i\}_{i=1}^{4n}$  completes the data required for the conceptual fitting problem in (1). While the problem is nonlinear and non-convex, this is typically not a problem because CAD data can provide an adequate initial guess for local solvers, such as Levenberg-Marquardt; see [19] for example.

### B. Our Contributions

Our contributions can be summarized as follows.

- (a) We introduce a novel method for estimating the rigid-body transform from target to LiDAR,  $H_T^L$ . For the point cloud pulled back to the origin of the LiDAR frame via the current estimate of the transformation, the cost is defined as the sum of the distance of a point to a 3D-reference target of known geometry<sup>1</sup> in the LiDAR frame, for those points landing outside of the reference target, and zero otherwise. We use an  $L_1$ -inspired norm in this work to define distance. Consistent with [18], we find that this is less sensitive to outliers than  $L_2$ -line fitting to edge points<sup>2</sup> using *ransac* [19].
- (b) In the above, by using the entire target point cloud when estimating the vertices, we avoid all together the delicate task of identifying the edge points and parsing them into individual edges of the target, where small numbers of points on some of the edges accentuate the quantization effects due to sparsity in a LiDAR point cloud.
- (c) We provide a round-robin validation study to compare our approach to a state-of-the-art method from 2018, namely [19]. A novel feature is the use of the camera image as a proxy for ground truth; in the context of 3D semantic mapping, this makes sense. A standard PnP method [17] is used to estimate the rigid-body transformation from LiDAR to camera. We also validate our algorithm on different numbers of targets.
- (d) Code for our method, our implementation of the baseline, and all the data sets used in this paper are released as open source; see [21].

<sup>1</sup>It has been given non-zero volume.

<sup>2</sup>Defined as the left-right end points of each LiDAR ring landing on the target.

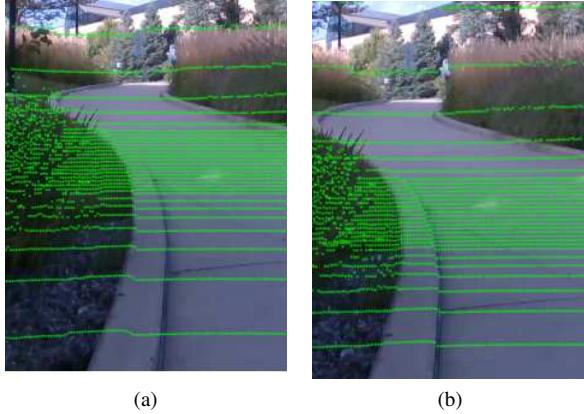


Fig. 3: (a) shows that a calibration result is not usable if it has few degrees of rotation error and a few percent of translation error. (b) shows good alignment of a LiDAR point cloud projected onto a camera image.

## II. FINDING THE LiDAR TARGET VERTICES

In this conference version of the work, we will assume that each target is planar, square, and rotated in the frame of the LiDAR by roughly  $45^\circ$  to form a diamond as in Fig. 2a. As indicated in [18], [19], placing the targets so that the rings of the LiDAR run parallel to its edges leads to ambiguity in the vertical position due to the spacing of the rings. We assume that standard methods have been used to automatically isolate the target's point cloud [22] and speak no further about it.

We take as a target's features its four vertices, with their coordinates in the LiDAR frame denoted  $\{X_i\}_{i=1}^4$ ; when useful, we abuse notation and pass from ordinary coordinates to homogeneous coordinates without noting the distinction. The vertices  $\{X_i\}_{i=1}^4$  are of course not directly observable in the point cloud. This section will provide a new method for estimating their coordinates in the frame of the LiDAR using an  $L_1$ -like norm.

### A. Remarks on LiDAR Point Clouds

LiDARs have dense regions and sparse regions along the z-axis, as shown in Fig. 2c. For a 32-beam Velodyne Ultra Puck, we estimate the resolution along the z-axis is  $0.33^\circ$  and  $1.36^\circ$  in the dense and sparse regions, respectively. A point's distance resolution along the y-axis is roughly  $0.1^\circ$ . The quantization error could be roughly computed from:  $d \sin \theta$  when a point is at  $d$  meter away. As a result, the quantization error could get quite large if one place the tag at a far distance. Figure 3a shows that a  $1^\circ$  of rotation error on each axis and 5% error in translation can significantly degrade a calibration result. As noted in [18], [19], it is essential to place a target at an appropriate angle so that known geometry can mitigate quantization error in the y- and z-axes.

### B. New Method for Determining Target Vertices

Let  $\mathcal{PC}$  denote the LiDAR target point cloud and let the collection of 3D points be  $\mathcal{X}_i$  so that  $\mathcal{PC} = \{\chi_i\}_{i=1}^N$ . The objective is to fit a reference target with vertices  $\{\bar{X}_i\}_{i=1}^4$ , defined as in Fig. 2e, onto the point cloud with “minimum

error”. It is actually easier to pull the  $\mathcal{PC}$  to the origin of the LiDAR through the current estimate of the inverse transformation  $H_L^L := (H_L^T)^{-1}$  and measure the error there.

**Remark 1.** To be clear, what we seek is a “best estimate” of the target vertices in the LiDAR frame and not the transformation itself. Our method is indirect because from the point cloud, we estimate a “best” LiDAR to target transformation, call it  $H_L^{T*}$ , and use it to define the vertices

$$X_i^* := H_L^{T*}(\bar{X}_i), \quad 1 \leq i \leq 4. \quad (2)$$

The correspondences of the estimated vertices with the physical top, bottom, and left or right sides of the target are not needed at this point. ■

For  $a \geq 0$  and  $\lambda \in \mathbb{R}$ , define

$$c(\lambda, a) := \begin{cases} \min\{|\lambda - a|, |\lambda + a|\} & \text{if } |\lambda| > a \\ 0 & \text{otherwise} \end{cases}; \quad (3)$$

see also the “soft  $L_1$  norm” in [18]. Let  $\{\tilde{\chi}_i\}_{i=1}^N := H_T^L(\mathcal{PC}) := \{H_T^L(\chi_i)\}_{i=1}^N$  denote the pullback of the point cloud by  $H_T^L$ , and denote a point's  $(x, y, z)$ -entries by  $(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i)$ . The cost is defined as

$$C(H_T^L(\mathcal{PC})) := \sum_{i=1}^N c(\tilde{x}_i, \epsilon) + c(\tilde{y}_i, d/2) + c(\tilde{z}_i, d/2), \quad (4)$$

where  $\epsilon \geq 0$  is a tuning parameter to account for uncertainty in the depth measurement of the planar target; see Fig. 2e.

We propose to determine the optimal  $H_T^L$  by

$$H_T^{L*} := \arg \min_{L \tilde{R}_{T,L}^L \tilde{T}_T^L} C(H_T^L(\mathcal{PC})). \quad (5)$$

Once the transformation is determined, the target vertices are defined by  $X_i := (H_T^{L*})^{-1}(\bar{X}_i)$ ,  $i = 1, \dots, 4$ , as in Remark 1.

**Remark 2.** It is underlined that the target vertices are being determined without extraction of edge points and their assignment to a side of the target. The cost in (4) treats the target as a rectangular volume.

## III. IMAGE PLANE CORNERS AND CORRESPONDENCES WITH THE LiDAR VERTICES

For a given camera image of a LiDAR target, let  $\{{}_C Y_i\}_{i=1}^4$  denote the vertices of the camera image. We assume that these have been obtained through the user's preferred method, such as corner detection [23], [24], [25], edge detection [26], [27], [28], or even manual selection. This is not the hard part of the calibration problem. To achieve simple correspondences  $X_i \leftrightarrow {}_C Y_i$ , the order of the indices on  $\{X_i\}_{i=1}^4$  may need to be permuted; we use the vertical and horizontal positions to sort them.

Once we have the correspondences, the next step is to project the vertices of the LiDAR target,  $[x_i \ y_i \ z_i \ 1]^T = X_i$ , into the image coordinates. The standard relations are [29], [30]

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{1}_{3 \times 3} \\ \mathbf{0}_{1 \times 3} \end{bmatrix}^T \begin{bmatrix} R_L^C & T_L^C \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \quad (6)$$

$${}_L Y_i = [u \ v \ 1]^T = \left[ \frac{u'}{w'} \ \frac{v'}{w'} \ 1 \right]^T, \quad (7)$$

where (6) includes the camera's intrinsic parameters and the extrinsic parameters ( $R_L^C, T_L^C$ ) that we seek.

For later use, we combine (6) and (7) to define

$$\Pi(X_i; R_L^C, T_L^C) := {}_L Y_i, \quad (8)$$

the projection map from LiDAR coordinates to image coordinates. Note that it is a function of both the extrinsic variables and the LiDAR point cloud.

#### IV. EXTRINSIC TRANSFORMATION OPTIMIZATION

This section assumes the vertices of the target in the LiDAR frame and in the camera's image plane have been determined, along with their correspondences. The optimization for the extrinsic transformation can be formulated in a standard PnP problem: minimize Euclidean distance of the corresponding corners. We also propose maximizing the intersection over union (IoU) of the corresponding projected polygons.

##### A. Euclidean distance

The standard PnP formulation is

$$(R_L^{C*}, T_L^{C*}) := \arg \min_{R, T} \sum_{i=1}^{4n} \|\Pi(X_i; R, T) - {}_C Y_i\|_2^2 \quad (9)$$

$$= \arg \min_{R, T} \sum_{i=1}^{4n} \|{}_L Y_i - {}_C Y_i\|_2^2, \quad (10)$$

where  ${}_C Y_i \in \mathbb{R}^2$  are the camera corners,  ${}_L Y_i \in \mathbb{R}^2$  are defined in (8), and  $n$  is the number of target poses.

##### B. IoU optimization

To compute the IoU of two projected polygons of the targets in an image plane, we define the intersection as a polygon with known coordinates<sup>3</sup>. We sort the vertices of the polygon counterclockwise using Graham scan, a method to find the convex hull of a finite set of points on a plane [31], [32]. The intersection area of the polygon ( $\mathbf{A}$ ) can be calculated via the Shoelace algorithm [33]:

$$\mathbf{A} = \frac{1}{2} \left| \sum_{i=1}^N \det \begin{bmatrix} x_i & x_{i+1} \\ y_i & y_{i+1} \end{bmatrix} \right|, \quad (11)$$

where  $x_{n+1} = x_1, x_0 = x_n$  as well as  $y_{n+1} = y_1, y_0 = y_n$ . We use Venn diagram to solve the union of the two projected polygons of the targets given the intersection calculated above and the area of each projection polygon.

<sup>3</sup>The vertices of the polygon consist of the 2D corners and the 2D line intersections and thus can be computed efficiently.

## V. EXPERIMENTAL RESULTS

In this section, we extensively evaluate our proposed method on seven different scenes through a form of "cross-validation": in a round-robin fashion, we train on one or more scenes and then evaluate on the remaining scenes. The quantitative evaluation consists of computing pixel error per corner, where we take the image corners as ground truth. We also show qualitative validation results by projecting the LiDAR scans onto camera images; we include here as many as space allows, with more scenes and larger images available at [21].

### A. Data Collection

The scenes include both outdoor and indoor settings. Each scene includes two targets, one approximately 80.5 cm square and the other approximately 15.8 cm square, with the smaller target placed closer to the camera-LiDAR pair. We use an *Intel RealSense Depth Camera D435* and a *32-Beam Velodyne ULTRA Puck LiDAR*, mounted on an in-house designed torso for a Cassie-series bipedal robot [3]. From the CAD file, the camera is roughly 20 cm below the LiDAR and 10 cm in front of it. The angle of the camera is adjustable. Here, its "pitch", in the LiDAR frame, is approximately zero.

A scan consists of the points collected in one revolution of the LiDAR's 32 beams. The data corresponding to a single beam is also called a ring. For each scene, we collect approximately 10 s of synchronized data, resulting in approximately 100 pairs of scans and images.

For each target, five consecutive pairs of LiDAR scans and camera images are selected as a data set. For each data set, we apply two methods to estimate the vertices of the targets, a baseline and the method in Sect. II-B. For each data set, we apply two methods to estimate the vertices of the targets, a baseline and the method in Sect. II-B.

### B. Baseline Implementation for LiDAR Vertices

As a baseline, we use the method in [19]. Because an open-source implementation was not released, we built our own, attempting to be as faithful as possible to the described method.

For each scan, the large and small targets are individually extracted from background [22]. For each target and group of five scans, we compute the extracted point cloud's centroid and center the point cloud about the origin. SVD is then used to find the target's normal and to orthogonally project the point cloud onto the plane defined by the normal and the centroid. For each scan, and for each ring hitting the target, the left and right end points of the ring are selected and then associated with one of the four edges of the target. Lines are fitted to each collection of edge points using least-squares and *ransac*. The vertices are obtained as the intersections of the lines in the plane defined by the target's normal and centroid. The four vertices are then re-projected to 3D space, as in Fig. 2d.

### C. Camera Corners and Associations

The process of determining the target corners begins by clicking near them in the image. This is the only manual

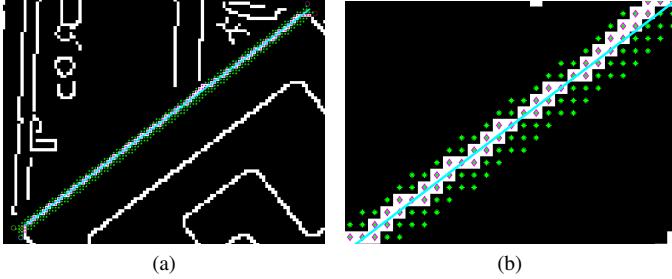


Fig. 4: (a) shows the result of edge detection. (b) shows the interior pixels (marked in green) of a bounding box given two clicked corners. The edge points and the edge line (as found by *ransac*) are marked in magenta and cyan, respectively. The corners are defined by the intersections of the resulting lines.

intervention required in the paper and even this process will soon be automated. As shown in Fig. 4a, between any given two clicked points, i.e., an edge of the target, a bounding box is drawn around the two points. Once we have roughly located the corners of the target, we process the image using the *edge* command in MATLAB with the ‘Canny’ option to detect the edge points within the bounding box. A line is then fit through each edge using *ransac*, and the intersections of the resulting lines define the target’s corners, as shown in Fig. 4b. A video and an implementation of this process is released along with the code.

#### D. Extrinsic Calibration

On the basis of the associated target vertices and camera-image corners, the PnP method of (9) is used to find the rigid-body transformation from LiDAR to camera. We also checked using the IoU (see Sec. IV-B), but because the results are similar, we are not reporting them here. The interested reader can consult [21]. SVD and *ransac* were computed using corresponding MATLAB commands, while the optimizations in (5) and (9) were done with *fmincon*.

#### E. Quantitative Results and Round-robin Analysis

In Table I, we present the RMS error of the LiDAR vertices projected into the camera’s image plane for the baseline<sup>4</sup>, labeled **RN**, and our method in (4) and (5), labeled **GL<sub>1</sub>**. In the case of two targets, a full round-robin study is performed: the rigid-body transformation from LiDAR to camera is “trained” on the combined set of eight vertices from both targets and then “validated” on the eight vertices of each of the remaining six scenes.

A complete round-robin study for four targets from two scenes would require 21 validations, while for six and eight targets, 35 validations each would be required. For space reasons, we report only a subset of these possibilities.

To be crystal clear, we are reporting in units pixel per corner

$$\sqrt{\frac{1}{4n} \sum_{i=1}^{4n} \|_L Y_i - {}_C Y_i\|_2^2}, \quad (12)$$

<sup>4</sup>SVD to extract the normal and *ransac* for individual line fitting, with the vertices obtained as the intersections of the lines.

where  $4n$  is the total number of target corners. In the case of two targets and one scene,  $n = 2$ . In the case of six targets from three scenes,  $n = 6$ . Figure 5 illustrates several point clouds projected to the corresponding camera images. Summary data is presented in Table II.

## VI. QUALITATIVE RESULTS AND DISCUSSION

In LiDAR to camera calibration, due to the lack of ground truth, it is traditional to show projections of LiDAR point clouds onto camera images. Often it is unclear if one is viewing training data or validation data. In Figure 6, we show a set of projections of LiDAR point clouds onto camera images for validation data. An additional set of images can be found in [21]. All of them show that the key geometric features in the image and point cloud are well aligned. The quality of the alignment has allowed our laboratory to build a high-quality (real-time) 3D semantic map with our bipedal robot Cassie Blue [34]. The map fuses LiDAR, camera, and IMU data; with the addition of a simple planner, it led to autonomous navigation [35].

Tables I and II show that **GL<sub>1</sub>** outperforms the baseline: on average, there is more than a 50% reduction in projection error and a 70% reduction in its variance. As for the sources of this improvement, we highlight the following points:

- (a) Least-squares-based methods estimate a normal vector for the target’s point cloud. As shown in Fig. 2a and Fig. 2b, even though the target is flat and has a well-defined normal, the returns in the LiDAR point cloud do not lie on a plane. Hence, a global normal vector does not exist.
- (b) Least-squares-based methods extract the target edge points from the point cloud for use in line fitting. The line fitting must be done in the plane defined by the estimated normal because, in 3D, non-parallel lines do not necessarily intersect to define a vertex.
- (c) **GL<sub>1</sub>** explicitly uses the target geometry in formulating the cost function. By assigning zero cost to interior points in the “ideal target volume” and non-zero otherwise, the optimization problem is focused on orienting the boundary of the target volume within the 3D point cloud. This perspective seems not to have been used before.
- (d) Hence, our approach does not require the (tedious and error-prone) *explicit* extraction and assignment of points to target edges. The determination of *boundary points* in 3D is *implicitly* being done with the cost function.

At the present time, our extrinsic calibration method is not “automatic”. The one manual intervention, namely clicking on the approximate target corners in the camera image, will be automated soon.

We have not conducted a study on how to place the targets. This is an interesting piece of future work because of the nonlinear operation required when projecting LiDAR points to the camera plane; see (6) and (7).

TABLE I: Training and validation data. The gray boxes denote the training set and the white boxes contain validation data. The numbers are the RMS errors of the LiDAR vertices projected to the image plane, measured in units of pixels per corner; see (12). For two targets, a complete round-robin study was done. For more targets, the number of combinations became prohibitive. S<sub>1</sub> through S<sub>7</sub> denote scene (experiment) number. The mean and standard deviation of each row—excluding the training set—are given in the last two columns.

Training\Validation	Method	# Tag	S1	S2	S3	S4	S5	S6	S7	mean	std
S1	RN	2	2.6618	8.7363	4.4712	7.3851	4.1269	7.1884	11.9767	7.3141	2.8991
	GL <sub>1</sub>	2	0.7728	<b>2.3303</b>	<b>2.3230</b>	<b>1.6318</b>	<b>1.3694</b>	<b>1.9637</b>	<b>2.7427</b>	<b>2.0602</b>	<b>0.5055</b>
S2	RN	2	3.3213	7.6645	4.9169	5.2951	4.0811	4.4345	7.7397	4.9648	1.5215
	GL <sub>1</sub>	2	<b>2.1368</b>	1.5181	<b>3.2027</b>	<b>3.2589</b>	<b>2.4480</b>	<b>4.1563</b>	<b>4.4254</b>	<b>3.2713</b>	<b>0.9039</b>
S3	RN	2	4.9909	9.5620	3.6271	8.7533	4.6421	10.1308	15.7764	8.9759	4.0654
	GL <sub>1</sub>	2	<b>4.6720</b>	<b>5.9350</b>	1.8469	<b>6.9331</b>	<b>4.4352</b>	<b>8.9493</b>	<b>15.3862</b>	<b>7.7185</b>	<b>4.1029</b>
S4	RN	2	21.2271	22.1641	17.5779	2.9452	15.9909	8.8797	15.3636	16.8672	4.7833
	GL <sub>1</sub>	2	<b>4.3681</b>	<b>4.7176</b>	<b>4.4804</b>	0.4986	<b>3.9004</b>	<b>3.3891</b>	<b>3.7440</b>	<b>4.0999</b>	<b>0.5040</b>
S5	RN	2	3.4621	8.3131	4.8500	7.6217	3.5197	7.4838	12.4364	7.3612	3.1066
	GL <sub>1</sub>	2	<b>2.0590</b>	<b>2.9541</b>	<b>3.0224</b>	<b>3.5483</b>	0.7392	<b>3.1386</b>	<b>6.0885</b>	<b>3.4685</b>	<b>1.3733</b>
S6	RN	2	29.4400	27.5404	27.9955	9.7005	20.6511	1.4253	9.5050	20.8054	9.1941
	GL <sub>1</sub>	2	<b>5.1207</b>	<b>5.0574</b>	<b>5.3537</b>	<b>2.0539</b>	<b>4.1739</b>	1.0012	<b>2.4194</b>	<b>4.0298</b>	<b>1.4503</b>
S7	RN	2	7.7991	9.9647	7.6857	4.1640	6.1619	2.3398	2.3708	6.3525	2.7512
	GL <sub>1</sub>	2	<b>2.1563</b>	<b>2.7837</b>	<b>3.1058</b>	<b>1.3234</b>	<b>2.4537</b>	<b>2.0838</b>	1.5389	<b>2.3178</b>	<b>0.6207</b>
S6-S7	RN	4	6.9426	9.6281	6.9136	3.9650	5.6420	2.2399	2.2399	6.6183	2.0764
	GL <sub>1</sub>	4	<b>2.2409</b>	<b>2.5607</b>	<b>2.9773</b>	<b>1.8215</b>	<b>2.1995</b>	0.3417	0.3417	<b>2.3600</b>	<b>0.4334</b>
S4-S6	RN	4	4.2476	8.5054	4.6712	2.8686	4.3311	2.8686	2.8748	4.9260	2.1153
	GL <sub>1</sub>	4	<b>1.0746</b>	<b>1.8871</b>	<b>2.3619</b>	0.2341	<b>1.5350</b>	0.2341	<b>2.6191</b>	<b>1.8955</b>	<b>0.6215</b>
S5-S6	RN	4	2.9309	8.0801	4.1196	3.4985	3.1519	3.1519	3.2204	4.3699	2.1201
	GL <sub>1</sub>	4	<b>1.1541</b>	<b>2.1026</b>	<b>2.5017</b>	<b>1.4267</b>	0.3291	0.3291	<b>2.5693</b>	<b>1.9509</b>	<b>0.6361</b>
S2-S5	RN	4	3.1991	6.0812	4.7311	5.4558	6.0812	4.8344	8.6576	5.3756	2.0139
	GL <sub>1</sub>	4	<b>0.9538</b>	0.4803	<b>2.3810</b>	<b>1.4556</b>	0.4803	<b>1.1592</b>	<b>2.5277</b>	<b>1.6955</b>	<b>0.7172</b>
S2-S3	RN	4	2.8820	6.2167	6.2167	4.4672	3.9536	3.7802	6.2324	4.2631	1.2406
	GL <sub>1</sub>	4	<b>1.0896</b>	0.4826	0.4826	<b>1.5349</b>	<b>1.5938</b>	<b>1.5597</b>	<b>2.7543</b>	<b>1.7065</b>	<b>0.6209</b>
S1-S7	RN	4	2.8595	8.2471	4.3297	3.7712	4.1023	2.4679	2.8595	4.5836	2.1710
	GL <sub>1</sub>	4	1.3542	<b>1.9973</b>	<b>2.3987</b>	<b>1.5926</b>	<b>1.4978</b>	<b>1.6130</b>	1.3542	<b>1.8199</b>	<b>0.3757</b>
S1-S6	RN	4	2.5633	8.0573	4.1762	3.6859	3.8973	2.5633	3.2569	4.6147	1.9535
	GL <sub>1</sub>	4	0.9753	<b>1.9154</b>	<b>2.3361</b>	<b>1.2906</b>	<b>1.2827</b>	0.9753	<b>2.3205</b>	<b>1.8291</b>	<b>0.5231</b>
S4-S5-S6	RN	6	2.9824	8.0943	4.1273	3.2821	3.2821	3.2821	3.0841	4.5720	2.4045
	GL <sub>1</sub>	6	<b>0.9713</b>	<b>1.9887</b>	<b>2.4395</b>	0.3288	0.3288	0.3288	<b>2.4369</b>	<b>1.9591</b>	<b>0.6918</b>
S2-S3-S5	RN	6	2.7816	5.5382	5.5382	4.5817	5.5382	3.8262	6.6229	4.4531	1.6239
	GL <sub>1</sub>	6	<b>1.0022</b>	0.4712	0.4712	<b>1.4755</b>	0.4712	<b>1.1722</b>	<b>2.7791</b>	<b>1.6073</b>	<b>0.8054</b>
S1-S3-S7	RN	6	3.3906	8.1677	3.3906	3.6688	3.9558	2.2913	3.3906	4.5209	2.5374
	GL <sub>1</sub>	6	1.7666	<b>1.9994</b>	1.7666	<b>1.5854</b>	<b>1.4754</b>	<b>1.5124</b>	1.7666	<b>1.6432</b>	<b>0.2419</b>
S2-S3-S4	RN	6	2.9428	5.5357	5.5357	5.5357	3.8913	2.4828	3.7673	3.2711	0.6733
	GL <sub>1</sub>	6	<b>0.9523</b>	1.8195	1.8195	1.8195	<b>1.5275</b>	<b>1.3631</b>	<b>2.4347</b>	<b>1.5694</b>	<b>0.6255</b>
S1-S2-S3	RN	6	5.3281	5.3281	5.3281	4.5086	3.8577	3.7165	6.4413	4.6310	1.2552
	GL <sub>1</sub>	6	1.7755	1.7755	1.7755	<b>1.3698</b>	<b>1.4807</b>	<b>1.3422</b>	<b>2.3156</b>	<b>1.6271</b>	<b>0.4629</b>
S5-S6-S7	RN	6	3.2743	8.2067	4.2628	3.5584	3.0628	3.0628	3.0628	4.8256	2.2921
	GL <sub>1</sub>	6	<b>0.9594</b>	<b>2.0149</b>	<b>2.3877</b>	<b>1.3754</b>	1.4905	1.4905	1.4905	<b>1.6843</b>	<b>0.6390</b>
S1-S3-S5-S7	RN	8	3.5316	8.1362	3.5316	3.4616	3.5316	2.2642	3.5316	4.6207	3.1029
	GL <sub>1</sub>	8	1.6904	<b>1.9886</b>	1.6904	<b>1.4743</b>	1.6904	<b>1.4136</b>	1.6904	<b>1.6255</b>	<b>0.3159</b>
S1-S2-S5-S7	RN	8	4.9240	4.9240	4.2049	3.6159	4.9240	2.2928	4.9240	3.3712	0.9793
	GL <sub>1</sub>	8	1.5269	1.5269	<b>2.4070</b>	<b>1.4788</b>	1.5269	<b>1.4650</b>	1.5269	<b>1.7836</b>	<b>0.5399</b>
S1-S3-S4-S5	RN	8	3.6113	8.0593	3.6113	3.6113	3.6113	2.4532	3.8397	4.7841	2.9199
	GL <sub>1</sub>	8	1.4986	<b>2.0563</b>	1.4986	1.4986	1.4986	<b>1.2256</b>	<b>2.6797</b>	<b>1.9872</b>	<b>0.7295</b>
S2-S4-S5-S7	RN	8	3.0803	5.0057	4.1893	5.0057	5.0057	2.3307	5.0057	3.2001	0.9351
	GL <sub>1</sub>	8	<b>0.9530</b>	1.5921	<b>2.4280</b>	1.5921	1.5921	<b>1.4865</b>	1.5921	<b>1.6225</b>	<b>0.7469</b>

TABLE II: A summary of validation data in Table I. This table compares mean and standard deviation for baseline and our approach as a function of the number of targets used in training. Units are pixel per corner.

	# Tag	2	4	6	8
RN	mean	10.3773	4.9645	4.3789	3.9940
GL <sub>1</sub>	mean	<b>3.8523</b>	<b>1.8939</b>	<b>1.6817</b>	<b>1.7547</b>
RN	std	7.0887	1.9532	1.7771	2.0467
GL <sub>1</sub>	std	<b>2.4155</b>	<b>0.5609</b>	<b>0.5516</b>	<b>0.5419</b>

## VII. CONCLUSIONS

We proposed a new method to determine the extrinsic calibration of a LiDAR-camera pair. When evaluated against a state-of-the-art baseline, it resulted in, on average, more than a 50% reduction in LiDAR-to-camera projection error and a 70% reduction in its variance. These results were established through a round-robin validation study when two targets are used in training, and further buttressed with results for training on four, six, and eight targets.

Two other benefits of our  $L_1$ -based method are: (1) it does not require the estimation of a target normal vector from an inherently noisy point cloud; and (2) it also obviates the identification of edge points and their association with specific sides of a target. In combination with lower RMS error and variance, we believe our results may provide an attractive alternative to current target-based methods for extrinsic calibration.

## ACKNOWLEDGMENT

Funding for this work was provided by the Toyota Research Institute (TRI) under award number N021515. Funding for J. Grizzle

was in part provided by TRI and in part by NSF Award No. 1808051. This article solely reflects the opinions and conclusions of its authors and not the funding entities. Dr. M Ghaffari offered advice during the course of this project. The first author thanks Wonhui Kim for useful conversations and Ray Zhang for generating the semantic image.

## REFERENCES

- [1] L. Gan, R. Zhang, J. W. Grizzle, R. M. Eustice, and M. Ghaffari, “Bayesian spatial kernel smoothing for scalable dense semantic mapping,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 790–797, April 2020.
- [2] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, “Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway,” in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4559–4566.
- [3] J. Huang. (2019) Cassie Blue walks Around the Wavefield. <https://youtu.be/LhFC45jweFMc>.
- [4] F. M. Mirzaei, D. G. Kottas, and S. I. Roumeliotis, “3D LiDAR-camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization,” *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 452–467, 2012.
- [5] X. Gong, Y. Lin, and J. Liu, “3D LiDAR-camera extrinsic calibration using an arbitrary trihedron,” *Sensors*, vol. 13, no. 2, pp. 1902–1918, 2013.
- [6] A. Dhall, K. Chelani, V. Radhakrishnan, and K. M. Krishna, “LiDAR-camera calibration using 3D-3D point correspondences,” *arXiv preprint arXiv:1705.09785*, 2017.
- [7] 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,” *arXiv preprint arXiv:1904.12433*, 2019.
- [8] J. Jiao, Q. Liao, Y. Zhu, T. Liu, Y. Yu, R. Fan, L. Wang, and M. Liu, “A novel dual-LiDAR calibration algorithm using planar surfaces,” *arXiv preprint arXiv:1904.12116*, 2019.
- [9] E.-S. Kim and S.-Y. Park, “Extrinsic calibration of a camera-LiDAR multi sensor system using a planar chessboard,” in *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2019, pp. 89–91.

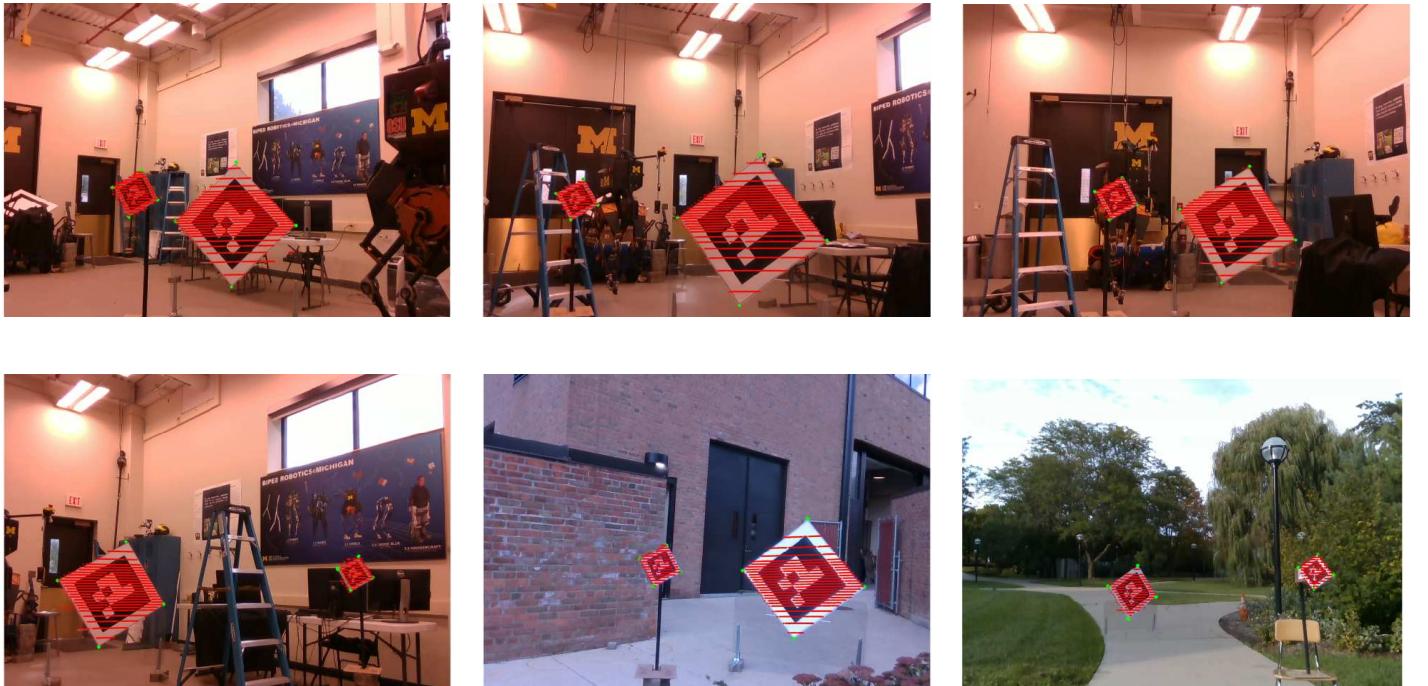


Fig. 5: Visual depiction of the validation data in the last row of Table I. For the method **GL<sub>1</sub>**, five sets of estimated LiDAR vertices for each target have been projected into the image plane and marked in green, while the target’s point cloud has been marked in red. Blowing up an image allows the numbers reported in the table to be visualized. The vertices are key.

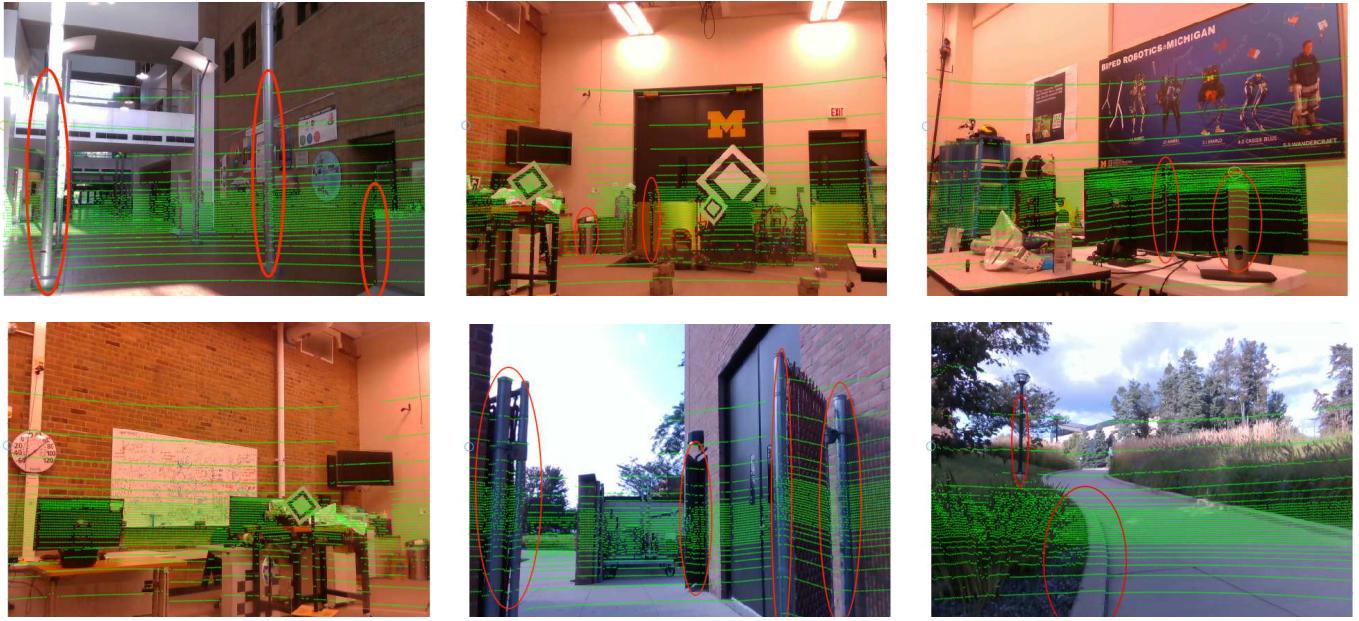


Fig. 6: Qualitative validation results. For the method **GL1** trained on S<sub>1</sub>, the LiDAR point cloud has been projected into the image plane for the other data sets and marked in green. The red circles highlight various poles, door edges, desk legs, monitors, and sidewalk curbs where the quality of the alignment can be best judged. The reader may find other areas of interest. Enlarge in your browser for best viewing.

- [10] C. Guindel, J. Beltrán, D. Martín, and F. García, “Automatic extrinsic calibration for LiDAR-stereo vehicle sensor setups,” in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.
- [11] A.-I. García-Moreno, J.-J. Gonzalez-Barbosa, F.-J. Ornelas-Rodriguez, J. B. Hurtado-Ramos, and M.-N. Primo-Fuentes, “LiDAR and panoramic camera extrinsic calibration approach using a pattern plane,” in *Mexican Conference on Pattern Recognition*. Springer, 2013, pp. 104–113.
- [12] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, “Automatic targetless extrinsic calibration of a 3D LiDAR and camera by maximizing mutual information,” in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.
- [13] Z. Taylor and J. Nieto, “Automatic calibration of LiDAR and camera images using normalized mutual information,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013.
- [14] J. Jeong, Y. Cho, and A. Kim, “The road is enough! extrinsic calibration of non-overlapping stereo camera and LiDAR using road information,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2831–2838, 2019.
- [15] J. Jiang, P. Xue, S. Chen, Z. Liu, X. Zhang, and N. Zheng, “Line feature based extrinsic calibration of LiDAR and camera,” in *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. IEEE, 2018, pp. 1–6.
- [16] W. Zhen, Y. Hu, J. Liu, and S. Scherer, “A joint optimization approach of LiDAR-camera fusion for accurate dense 3D reconstructions,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3585–3592, 2019.
- [17] V. Lepetit, F. Moreno-Noguer, and P. Fua, “Epnp: An accurate o (n) solution to the pnp problem,” *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.
- [18] Q. Liao, Z. Chen, Y. Liu, Z. Wang, and M. Liu, “Extrinsic calibration of LiDAR and camera with polygon,” in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2018, pp. 200–205.
- [19] L. Zhou, Z. Li, and M. Kaess, “Automatic extrinsic calibration of a camera and a 3D LiDAR using line and plane correspondences,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5562–5569.
- [20] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, “Contact-aided invariant extended kalman filtering for robot state estimation,” *The International Journal of Robotics Research*, p. 0278364919894385, 2019.
- [21] J.K. Huang and Jessy W. Grizzle, “Extrinsic LiDAR Camera Calibration,” 2019. [Online]. Available: [https://github.com/UMich-BipedLab/extrinsic\\_lidar\\_camera\\_calibration](https://github.com/UMich-BipedLab/extrinsic_lidar_camera_calibration)
- [22] J.-K. Huang, M. Ghaffari, R. Hartley, L. Gan, R. M. Eustice, and J. W. Grizzle, “LiDARTag: A real-time fiducial tag using point clouds,” *arXiv preprint arXiv:1908.10349*, 2019.
- [23] C. G. Harris, M. Stephens, et al., “A combined corner and edge detector.” in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.
- [24] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *European conference on computer vision*. Springer, 2006, pp. 430–443.
- [25] E. Rosten, R. Porter, and T. Drummond, “Faster and better: A machine learning approach to corner detection,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 1, pp. 105–119, 2008.
- [26] J. Canny, “A computational approach to edge detection,” in *Readings in computer vision*. Elsevier, 1987, pp. 184–203.
- [27] J. S. Lim, “Two-dimensional signal and image processing,” *Englewood Cliffs, NJ, Prentice Hall*, 1990, 710 p., 1990.
- [28] O. R. Vincent, O. Folorunso, et al., “A descriptive algorithm for sobel image edge detection,” in *Proceedings of Informing Science & IT Education Conference (InSITE)*, vol. 40. Informing Science Institute California, 2009, pp. 97–107.
- [29] R. Hartley and A. Zisserman, “Multiple view geometry in computer vision second edition,” *Cambridge University Press*, 2000.
- [30] D. A. Forsyth and J. Ponce, *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.
- [31] R. L. Graham, “An efficient algorithm for determining the convex hull of a finite planar set,” *Info. Pro. Lett.*, vol. 1, pp. 132–133, 1972.
- [32] A. M. Andrew, “Another efficient algorithm for convex hulls in two dimensions,” *Information Processing Letters*, vol. 9, no. 5, pp. 216–219, 1979.
- [33] B. Braden, “The surveyor’s area formula,” *The College Mathematics Journal*, vol. 17, no. 4, pp. 326–337, 1986.
- [34] L. Gan, R. Zhang, J. W. Grizzle, R. M. Eustice, and M. Ghaffari, “Bayesian spatial kernel smoothing for scalable semantic mapping,” *arXiv preprint arXiv:1909.04631*, 2019.
- [35] “Autonomous Navigation and 3D Semantic Mapping on Bipedal Robot Cassie Blue,” <https://youtu.be/N8THn5YGxPw>, accessed: 2020-01-31.