Improvements to Target-Based 3D LiDAR to Camera Calibration

Jiunn-Kai Huang and Jessy W. Grizzle

Abstract—The homogeneous 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 translation 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 inherent in a LiDAR image of a target, ameliorate target pose estimation in face of the quantization errors in their distance measurements. This paper proposes (1) the use of targets of known dimension and geometry to overlaying a LiDAR image on a camera image. The biggest lead to 20 cm translation errors at a distance of 5 m when of a few degrees in rotation or a few percent in translation can be considered glamorous in any sense of the word, it is nonetheless such as SLAM. While determining such a transformation is not crucial for many modern autonomous systems, DAR and monocular camera is required for sensor fusion tasks, thus an optimization problem of the form

\[ \min_{R,T} \text{dist}(P(H_L^C(X_i), Y_i)) \]

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

I. INTRODUCTION AND RELATED WORK

The desire to produce 3D-semantic maps with our Cassie-series bipedal robot presented in [1] and [2] has motivated us to fuse 3D-LiDAR and RGB-D monocular camera data for improved navigation. 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.

In this paper, we assume that the intrinsic calibration of the two sensors has already been done [3] and focus on obtaining the rigid transformation, i.e. rotation matrix and translation, between a LiDAR and a camera. This is a well studied problem with a rich literature that can be roughly divided into methods that require targets: [4], [5], [6], [7], [8], [9], [10] and those that do not: [11], [12], [13], [14], [15].

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. If “enough” independent correspondences can be made, the LiDAR to camera transformation can be found by Perspective-n-Point (PnP) as in [16], which is, though an optimization problem of the form

\[ (R^*_L, T^*_L) = \arg \min_{(R,T)} \text{dist}(P(H_L^C(X_i), Y_i)) \]

The works closest to ours are [17], [18]. 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 [17] and Fig. 1 in [18]. More generally, we recommend the literature overview in [17] 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 to use of the SVD to find the normal vector and more broadly to least squares line fitting problems [18], while (a) opens up other perspectives, as highlighted in [17].

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 \( \{Y_i\}_{i=1}^n \) 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 [18] for example.

### B. Our Contributions

Our contributions can be summarized as follows.

(a) We make use of the target’s geometry when estimating its vertices. In particular, if the target is a diamond, vertices form right angles and the lengths of the sides are equal. We show that this simple change improves the accuracy of estimating the vertices. Essentially, the extra constraints allow all of the target’s boundary points to be pulled into the regression problem at once, instead of parsing them into individual edges of the target, where small numbers of points on some of the edges will accentuate the quantization effects due to sparsity in a LiDAR point cloud.

(b) We introduce a novel method for estimating the rigid body transform from target to LiDAR, \( H^T \). For the point cloud pulled back to 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\(^1\) 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. As in [18], we find that this is less sensitive than using the \( L_2 \)-norm.

(c) We exploit the fact that the variance of the camera data is significantly lower than that of the LiDAR data. Hence, after the LiDAR to camera transformation has been estimated by solving the PnP problem, we introduce a refinement step to the estimated pose of the LiDAR target given the current estimate of the LiDAR to camera transformation. In this process, the modified vertices remain consistent with the target geometry and the target’s point cloud.

(d) We provide a round-robin validation study to support our approach. Once again, we place our faith in the camera image and use it as a proxy for ground truth. In the context of 3D semantic mapping, this makes sense.

(e) Our code is released as open source.

### 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° to form a diamond as in Fig. 2a. As indicated in [17], [18], 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 [19] 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. It will also provide a simple improvement to existing methods that use least squares fitting.

### 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° and 1.36° in the dense and sparse regions, respectively. A point’s distance resolution along the y-axis is roughly 0.1°. Figure 3a shows that a 1° of rotation error on each axis and 5% error in translation can significantly degrade a calibration.
result. As noted in [17], [18], 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. To overcome quantization error in the $x$-axis, we accumulate a few scans to estimate a target’s pose and “condition on” the current point cloud, we estimate a “best” LiDAR to target transformation itself. Our method is indirect because from a tuning parameter to account for uncertainty of the target vertices in the LiDAR frame and not the target vertices in Fig. 2e, onto the point cloud with “minimum error”. It is actually easier to pull the PC in Fig. 2e, onto the point cloud with “minimum error”. It is straightforward to lift them to vertices $\bar{X}$, $\bar{Y}$, and use it to define the vertices $\bar{X}_i := H^T_{L} (\bar{X}_i)$, $1 \leq i \leq 4$. (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

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

see also the “soft $L_1$ norm” in [18]. Let $\{\tilde{x}_i\}_{i=1}^N := H^T_L (\{X_i\}_{i=1}^N \subset \mathbb{R}^3$ 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

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

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

We propose to determine the optimal $H^T_L$ by

$$H^T_L := \arg \min_{L, R^T_L T^T_L} C(H^T_L (\mathcal{P}C)). \quad (5)$$

Once the transformation is determined, the target vertices are defined by $X_i := (H^T_L)^{-1}(\tilde{X}_i), i = 1, \ldots, 4$, as in Remark 1.

C. Improvement to Least Squares Approaches for Finding the Target Vertices

We assume the target’s normal vector has been estimated using methods in [17], [18] and that the point cloud $\mathcal{P}C$ has been orthogonally projected to the plane of the target. In [17], [18], lines along the four edges of the projected target are regressed individually after appropriately parsing the edge points. For “robustness”, $\text{ransac}$ is typically used.

Instead, we propose to use all of the edge points simultaneously when regressing the edges. The basic formulation remains one of fitting four lines to the four sets of edge points, as is standard in the area,

$$z = m_i y + b_i, 1 \leq i \leq 4 \quad (6)$$

and then adding the constraints for a square. For opposite sides to be parallel, one needs

$$m_1 = m_3, \quad m_2 = m_4, \quad (7)$$

and to form right angles,

$$m_1 m_2 = -1. \quad (8)$$

The least squares line-fitting problem with (7) is a QP and already allows edges on opposite sides of the diamond to contribute to finding the slopes. While the constraint (8) links all the edges together, it makes the least squares problem non-convex. However, applying Taylor’s theorem about “current estimates” of the slopes at step $t$, denoted $m^t_1, m^t_2$, along with some simple algebra results in the linear (update) constraint

$$\begin{bmatrix} m^t_2 & m^t_1 \\ m^{t+1}_2 & m^{t+1}_1 \end{bmatrix} = -1 + m^t_1 m^t_2. \quad (9)$$

Starting with the initial estimates of $m^0_1 = 1, m^0_2 = -1$, in our experience, iterating the QP with linear constraints

$$A_{eq} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ m^t_2 & m^t_1 & 0 & 0 \end{bmatrix}, \quad b_{eq} = \begin{bmatrix} 0 \\ 0 \\ -1 + m^t_1 m^t_2 \end{bmatrix} \quad (10)$$

on the slopes, and no constraints on the offsets, converges in two or three steps. In principle, one should also impose conditions on the offsets $b_i$ so that the lengths of the sides are equal to the physical size of the target, but in our experience, it is not necessary to do this.

The intersections of the resulting lines define the target’s vertices in the plane defined by its normal vector. It is then straightforward to lift them to vertices $\{X_i\}_{i=1}^4 \subset \mathbb{R}^3$ of the target in $\mathbb{R}^3$. The reader is referred to [20] for the corresponding transformation, $H^T_L$.  

III. Vertices in the Image Plane and Correspondences with the LiDAR Vertices

For a given camera image of a LiDAR target, let $\{y_i\}_{i=1}^4$ denote the vertices of the camera image. We assume

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image.png}
\caption{(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 LiDAR point cloud on the image.}
\end{figure}
that these have been obtained through the user’s preferred method, such as corner detection [21], [22], [23], edge detection [24], [25], [26], or even manual selection. This is not the hard part of the calibration problem. To achieve simple correspondences \( X_i \leftrightarrow cY_i \), the order of the indices on \( \{X_i\}^4_{i=1} \) 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, \( x_i, y_i, z_i \) to \( X_i \), into the image coordinates. The standard relations are [27], [28]

\[
\begin{bmatrix}
  u'
  \\
  v'
\end{bmatrix} = 
\begin{bmatrix}
  f_x & s & c_x \\
  0 & f_y & c_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  R^C_L & T^C_L \\
  0_{1 \times 3} & 1
\end{bmatrix}
\begin{bmatrix}
  X_i \\
  1
\end{bmatrix},
\]

(11)

\[
X_i = [u' \ v' \ 1]^T = [u' \ w' \ w' \ 1]^T,
\]

(12)

where (11) includes the camera’s intrinsic parameters and the extrinsic parameters \( (R^C_L, T^C_L) \) that we seek.

For later use, we combine (11) and (12) to define

\[
\Pi(X_i; R^C_L, T^C_L) := X_i,
\]

(13)

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

IV. EXTRINSIC OPTIMIZATION AND POSE REFINEMENT

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.

A. Extrinsic Transformation Optimization

The optimization for the extrinsic transformation can be formulated in two ways: minimize Euclidean distance of the corresponding corners or maximize the intersection over union (IoU) of the corresponding polygons.

1) Euclidean distance: The standard PnP formulation is

\[
(R^*_L, T^*_L) := \arg \min_{R,T} \sum_{i=1}^{4n} ||\Pi(X_i; R, T) - cY_i||^2_2 \]

(14)

\[
= \arg \min_{R,T} \sum_{i=1}^{4n} ||lY_i - cY_i||^2_2,
\]

(15)

where \( cY_i \in \mathbb{R}^2 \) are the camera corners, \( lY_i \in \mathbb{R}^2 \) are defined in (13), and \( n \) is the number of target poses.

2) 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\(^2\). 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 [29], [30]. The intersection area of the polygon can be calculated via the Shoelace algorithm [31].

\(^2\)The vertices of the polygon consist of the 2D corners and the 2D line intersections and thus can be computed efficiently.

B. Refinement of the LiDAR Target Vertices

The above methods optimize \( (R^C_L, T^C_L) \) for given values of \( \{X_i\}^4_{i=1} \). Due to the systematic errors mentioned in Sec.II-A, we suspect the LiDAR vertices are less accurate than those of the camera’s image. We therefore propose a refinement of the vertices conditioned on the current estimate of the projection map, \( \Pi \), in (13), that is, based on the current estimate of the extrinsic parameters \( (R^C_L, T^C_L) \). Once the vertices are updated, new estimates for \( (R^C_L, T^C_L) \) can be obtained, giving us an alternating two-step process:

Step 1: Adjust vertices with current \( t^C_L, t^C_L \) held fixed

\[
\delta H := \begin{bmatrix}
  \delta R & \delta T \\
  0 & 1
\end{bmatrix}
\]

\[
\delta H^* := \arg \min_{\delta H} \sum_{i=1}^{4n} ||\Pi(\delta H \circ t^C_L, t^C_L) - cY_i||^2_2 \\
+ \rho C((H^T_L \circ \delta H)^{-1}(PC));
\]

(16)

Step 2: Refine calibration \( R^C_L, T^C_L \) using updated vertices

\[
t^{+1}X_i := \delta H^* \circ t^C_L \\
t^{+1}_L Y_i = \Pi(t^{+1}X_i; t^C_L, t^C_L),
\]

(17)

\[
t^{+1}(R^*_L, T^*_L) = \arg \min_{R,T} \sum_{i=1}^{4n} ||l^{+1}Y_i - cY_i||^2_2,
\]

where superscript \( t \) denotes iteration number, \( X_i \) and \( cY_i \) are as before, and the weight \( \rho \) trades off confidence in the point cloud vs the camera.

V. EXPERIMENTAL RESULTS

In this section, we extensively evaluate our proposed methods on seven different scenes though a form of “cross-validation”: in a round-robin fashion, we train on one scene and then evaluate on the other six. The quantitative evaluation consists of computing pixel error per corner, where we take the image vertices 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 [20].

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 [2]. 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. For each scene, we collect approximately 10 s of synchronized data, resulting in approximately 100 pairs of scans and images.

B. Data Processing

For each scene, 15 consecutive pairs of LiDAR scans and camera images are extracted and divided into 5 data sets.
For each data set, we apply three methods to estimate the vertices of the two targets:

- **RN:** Uses SVD to extract the normal and *ransac* for individual line fitting, with the vertices obtained as the intersections of the lines.
- **GN:** Uses SVD to extract the normal and the QP associated with (10) to fit a square.
- **GL₁:** Uses (5) and (2) to find the vertices.

For each of these, we include the refinement step of Sec. IV-B, and denote them by **RN-R**, **GN-R**, and **GL₁-R**. For **GL₁-R**, the cost term in (16) is given by (4), while for the two least squares approaches, we measure the Euclidean norm of the 3D-edge points of the point cloud to the lines defined by the refined target vertices.

In all of the above, the least-squares-based PnP method of (14) is being used to find the extrinsic calibration. We also checked using the IoU (see Sec. IV-A.2), but because the results are similar, we are not reporting them here. The interested reader can consult [20]. SVD, *ransac*, and QP were computed using standard MATLAB commands, while the optimizations in (5), (14), (16) and (17) were done with *fmincon*.

**C. Quantitative Results**

Once again, we are taking the camera data as a proxy for ground truth. In Table I, we compute the RMS error of the LiDAR vertices projected into the image plane for each of the above methods as shown in Fig. 4. To be crystal clear, we are reporting

\[
\sqrt{\frac{1}{4n} \sum_{i=1}^{4n} \| Y_i - c Y_i \|^2},
\]

where \(4n = 40\) is the total number of target corners. Hence, the units are pixel per corner.

**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 5, 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 [20]. All of them show that the key geometric features in the image and point cloud are well aligned and will allow us to achieve our goal of 3D semantic mapping with our bipedal robot Cassie Blue.

Table I shows that **GL₁** and **GL₁-R** outperform the other methods. We conjecture that **GL₁** outperforms the least-squares methods because it avoids estimating a normal vector for the target’s point cloud. The systematic variations in the target depth may be rendering the normal computation inaccurate. We conjecture that the refinement method did not provide significant improvement to **GL₁** because the errors were already so small.

Imposing geometry when fitting the edges improves the least-square methods. Moreover, refining the target vertices in the LiDAR point cloud on the basis of the camera data also proved useful. We emphasize these points because one may be able to use them to improve several current packages for extrinsic calibration. The main concerning point of the data is the variability in the least squares results.

Throughout the paper, we have avoided the term “automatic” in regards to our calibration method. Could it be applied? Perhaps the reviewers will have an opinion. We have released the MATLAB-version of our software with this submission. The only manual intervention required is, once the image edges have been detected using the *edge* command in MATLAB with the ’Canny’ option, we manually click the target corners to obtain their coordinates. This will be automated soon.

**VII. CONCLUSIONS**

We proposed improvements to target-based LiDAR-camera calibration. We evaluated our proposed methods and compared them with other approaches in a round-robin validation study. In the validation study, our \(L₁\)-based method achieved a per corner RMS error, measured in the image plane, on the order of a few pixels, when comparing the projected LiDAR vertices to the image corners. The validation results show our methods are repeatable and outperform other target-based methods.

**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.
TABLE I: Round-robin validation study. The gray box denotes the training set and the others are used as validation data. The numbers are the RMS errors of the LiDAR vertices projected to the image plane in pixels per corner; see (18).

| Training/Validation | Method | S1 | S2 | S3 | S4 | S5 | S6 | S7 |
|----------------------|--------|----|----|----|----|----|----|----|
| S1                   | RN     | 181.316 | 93.736 | 111.898 | 158.531 | 57.896 | 96.584 | 167.754 |
|                      | RN-R   | 182.352 | 84.828 | 102.918 | 138.973 | 43.536 | 83.368 | 155.562 |
|                      | GN     | 181.315 | 66.953 | 79.913 | 50.607 | 113.239 | 57.386 | 96.384 |
|                      | GN-R   | 182.252 | 60.586 | 73.497 | 47.483 | 99.270 | 48.019 | 85.930 |
|                      | GL     | 1.546 | 1.465 | 1.437 | 1.685 | 0.880 | 1.098 | 1.986 |
|                      | GL-R   | 1.443 | 1.383 | 1.398 | 1.590 | 0.775 | 0.918 | 1.772 |
| S2                   | RN     | 147.794 | 108.470 | 115.953 | 110.759 | 75.704 | 93.369 | 122.773 |
|                      | RN-R   | 142.731 | 111.257 | 118.645 | 110.603 | 77.741 | 92.488 | 122.332 |
|                      | GN     | 18.089 | 16.072 | 14.949 | 14.299 | 7.581 | 13.920 | 81.740 |
|                      | GN-R   | 16.210 | 12.468 | 15.037 | 14.099 | 7.181 | 9.343 | 19.752 |
|                      | GL     | 1.196 | 1.470 | 1.712 | 1.667 | 0.944 | 2.040 | 3.255 |
|                      | GL-R   | 1.021 | 1.322 | 1.589 | 1.465 | 0.823 | 1.772 | 2.781 |
| S3                   | RN     | 82.742 | 77.428 | 126.157 | 118.209 | 68.545 | 98.988 | 126.225 |
|                      | RN-R   | 83.433 | 68.193 | 99.310 | 90.853 | 58.459 | 76.674 | 108.076 |
|                      | GN     | 12.165 | 12.599 | 10.902 | 11.138 | 6.491 | 11.071 | 11.649 |
|                      | GN-R   | 11.948 | 12.325 | 9.851 | 10.924 | 5.657 | 10.826 | 10.661 |
|                      | GL     | 2.552 | 2.971 | 8.288 | 3.513 | 2.349 | 4.595 | 8.136 |
|                      | GL-R   | 2.880 | 2.308 | 6.115 | 2.580 | 1.750 | 3.346 | 8.996 |
| S4                   | RN     | 97.964 | 159.075 | 102.341 | 102.544 | 151.029 | 149.366 | 143.103 |
|                      | RN-R   | 69.939 | 118.571 | 72.065 | 101.823 | 108.226 | 104.654 | 122.885 |
|                      | GN     | 38.259 | 49.452 | 36.823 | 6.536 | 35.846 | 32.127 | 52.071 |
|                      | GN-R   | 37.189 | 48.717 | 35.937 | 5.224 | 35.268 | 31.653 | 31.421 |
|                      | GL     | 1.575 | 1.538 | 1.678 | 0.608 | 1.390 | 1.321 | 1.071 |
|                      | GL-R   | 1.494 | 1.472 | 1.623 | 0.564 | 1.319 | 1.303 | 1.075 |
| S5                   | RN     | 123.945 | 97.514 | 132.301 | 114.755 | 177.790 | 90.855 | 124.152 |
|                      | RN-R   | 112.834 | 86.316 | 119.455 | 100.701 | 177.249 | 83.311 | 110.439 |
|                      | GN     | 14.275 | 17.187 | 14.252 | 8.145 | 5.288 | 10.748 | 19.338 |
|                      | GN-R   | 14.252 | 17.025 | 14.219 | 8.046 | 4.425 | 10.535 | 18.914 |
|                      | GL     | 1.444 | 1.559 | 1.390 | 1.306 | 1.444 | 1.696 | 3.306 |
|                      | GL-R   | 1.421 | 1.793 | 1.396 | 1.754 | 1.038 | 1.748 | 3.394 |
| S6                   | RN     | 73.994 | 103.545 | 92.773 | 111.625 | 98.759 | 48.956 | 106.110 |
|                      | RN-R   | 49.349 | 67.209 | 54.542 | 58.689 | 59.899 | 47.769 | 68.004 |
|                      | GN     | 13.824 | 19.037 | 18.629 | 11.423 | 15.289 | 6.353 | 17.743 |
|                      | GN-R   | 11.873 | 17.725 | 12.174 | 7.562 | 9.299 | 2.752 | 9.462 |
|                      | GL     | 2.895 | 2.486 | 3.064 | 1.102 | 2.102 | 1.226 | 2.348 |
|                      | GL-R   | 2.717 | 2.351 | 2.912 | 1.087 | 2.004 | 1.142 | 2.272 |
| S7                   | RN     | 59.186 | 52.019 | 39.212 | 33.833 | 42.461 | 32.817 | 66.453 |
|                      | RN-R   | 57.494 | 72.173 | 57.880 | 51.469 | 65.713 | 54.498 | 86.117 |
|                      | GN     | 0.707 | 0.963 | 1.163 | 1.180 | 0.923 | 0.949 | 1.682 |
|                      | GN-R   | 0.692 | 0.936 | 1.139 | 1.144 | 0.899 | 0.933 | 1.616 |

Fig. 5: Qualitative validation results. For the method GL-R trained on S1, 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.
