An Iterative Camera-to-lidar Calibration method using two frames of object points

Ju Wang, Venkat Dasari, Billy Geerhart, Brian Rapp, Peng Wang

September 16, 2023

1 Introduction

There is a growing need for distributed robot systems to share and render 2D and 3D sensory data from different sources in an accurate and consistent manner. This capability is important for performing tasks such as collaborative exploration, mapping, search and rescue, and data collection. In general, multi-robot missions require sensory information to be shared between the agents for effective operation. Fully autonomous multi-agent exploration, in particular, requires distributed information sharing in several essential 'backend' tasks: (1) merging of the member robots' sub-maps into a global map, (2) identification and localization of interesting objects, and (3) visualization of 2D and 3D data. To support the above computing tasks, an efficient and robust algorithm to calibrate intra-sensor and inter-sensor parameters is required to correctly integrate data streams from multiple sources. Such algorithms, often implemented as a parametric estimation method, are critical to accurately "superimpose" synchronized data streams such as the camera data and the lidar point cloud.

A promising method to superimpose different sensor data is the diffusion technique [1], which applies rigid transformations between sensors to obtain an initial alignment, and then uses inherent local features to fine-tune the associations between the raw data. Many diffusion algorithms depend on an accurate camera-lidar transformation model. This raises a potential issue: slight errors in the transformation matrix can result in significant mislabeling of the sensor data. One widely used remedy is to perform constant re-calibration of the sensor TF matrix to minimize alignment-induced errors. To alleviate the need for frequent re-calibration and reduce the impact of the calibration error, we propose a self-calibration method that allows the robot to estimate the transform (TF) matrix with minimum human involvement and setup requirement. The inter-sensor calibration process typically estimates 6D relative pose or the equivalent transformation matrix between two sensors that are rigidly mounted. In our case, we use the lidar sensor and the camera sensor.

Our lidar-camera calibration method is an iterative search process that minimizes the 3D association error of the observed object at two observation events. Our method, referred to as Iterative Closest PnP (ICPnP), can be considered as an extension of the Perspective-n-Point algorithm but without the assumption of the 3D-2D association. The problem can be formulated as a bundle adjustment problem between discrete time camera data and lidar data from the same robot. We found that two LIDAR observations allow us to estimate the relative pose change of the robot using point cloud matching algorithms such as floam [2] and GICP [3], assuming moderate robot movement. Vice versa, this method can also be used to estimate the position change of a moving object and to segment the moving objects from the background in 3D domains. Together with the object pixels in the image plane, the estimation of the transformation parameters can thus be solved by finding the optimum 2D-to-3D correspondence of the target object.

Following these observations, our calibration method is summarized below:

1. Perform two observations of the dominant object separated by a movement displacement $(\Delta P, \Delta yaw)$
2. Compute the corresponding lidar points P(0) and P(1)
3. Compute the closet 2D image plane points from P(k)
4. Compute a new $[T|R]$ matrix based on current point correspondence
5. Apply the new transform to get the projected image points P(k+1)
6. Repeat the above steps until convergence

We evaluated the proposed method using field experimental data collected by a fully loaded Clearpath Jackal robot. Our implementation and computing pipeline is optimized to achieve real-time performance on resource-constrained edge computing autonomous platforms, e.g., unmanned ground vehicles (UGVs).

2 Related Work

The lidar-camera calibration is closely related to the classic Perspective-n-Point problems [4, 5, 6]. Camera pose estimation [4], also referred to as view point recover problem [5], uses 2D-3D correspondence to establish a transformation matrix between 3D observation and 2D observations. The three point P3P case and the four point P4P case, close-form solution can be obtained [4]. For $n > 4$, the problem must be solved via numerical methods. High accuracy correspondence between the 2D pixels and the 3D points perceived by the robot is essential to build high-level understanding of the world and reasoning in a symbolic world model.

A closely related problem is to estimate rigid motion using two sets of 3D point clouds. Since Besl's seminal ICP paper [7], there is a rich collection of improvements in recent decades [3], especially with the availability of high-resolution 360-degree lidar sensors. The main idea in ICP and its variant is to gradually recompute the point correspondence in such a way that a better transformation matrix can be computed. Our method borrows a similar strategy to assure the convergence of the solution.

The recent effort in this line of work is to use a deep neural network to build point correspondence. The problem however remains a significant challenge. In the 2D domain, Object detection and segmentation using 2D images have seen high-accurate solutions through the publication of deep convolutional neural networks models [8, 9].

In the 3D domain, there are fewer results due to many notable difficulties associated with the 3D point representation and sensor data. For one, the processing of 3D points is much less efficient than its 2D sibling due to its sparseness and wider dynamic range. The 3D point clouds are unstructured by nature and laborious to label, which results in less availability of training data. Recently there are some promising methods that have demonstrated the feasibility of converting point clouds as input for deep networks [10],

Our work is also inspired by the LDLS method [1]. The LDLS method uses an Mask-RCNN [9] to detect object classes and instances at the pixel level. The author demonstrates that the 2D label can be diffused to the 3D domain by repeatedly applying a connectivity matrix to the 2D labeling, which can be executed in a GPU with much-improved computing time. The LDLS method leverages the success of convolutional neural networks for 2D image segmentation, However, LDLS shows high segmentation error if lidar-camera calibration is not accurate. The error is particularly noticeable near the edge of objects where they board with background or occlude with other objects.

A key assumption in LDLS and other similar methods is that calibration parameters between the 2D and 3D sensors are well-established. Our contribution is the development of a semi-automatic method that allows the calibration to be done in a much unrestrictive environment by taking advantage of the 2D and 3D observations that are already available at the robot platform.

3 Problem Formulation

3.1 Calibration Parameter Estimation

The input to the classic PnP problem consists of a 3D point set P observed in the world frame and the corresponding image point set I observed in the camera image plane, and the solution is a rigid transformation from the world frame to the camera frame, A major difference in our problem setting is that we don't have the 2D-3D point correspondence information. Our method boils down to a novel method to obtain the 2D-3D correspondence, after which the problem can be easily solved using the existing PnP methods. Before diving into our method, it is necessary to establish some commonly used notations. Let P_w be the point set in 3D world frame, P_c be the coordinate at the camera frame, and I_o the projected pixel set in the image plane, the general calibration process is formulated as a non-linear optimization problem where we minimize the average

Table 1: Symbols definition

$P_c(t)$	Object Point cloud (PC) at t in camera frame
$P_l(t)$	True PC at t in robot lidar frame
(x_c, y_c, z_c)	3D point in camera frame
$\bar{s} = (x, y, z, \bar{q})$	object position and quaternion in lidar frame
$I_o(t)$	Object 2D image at t
$\hat{P_o^*}(t)$	predicted Object PC at t
TR_c	TF matrix to cam frame
M	2D cam proj matrix

corner errors. Essentially, we optimize

$$argmin_{T,R} \sum_{p \in P} ||d(TR(p), I_o)||$$

where $TR(p)$ is transformed coordinate of $p \in P_w$ under given calibration parameters TR, and $d(p, A)$ is the distance of p to a 2D point set.

The parameter set to be estimated is the transformation matrix $[T|R]$. We assume that the camera projection matrix M is known. The 3D transformation matrix $TR_{lc} = [R\ |T]$ is break to two parts: R is a 3x3 rotation matrix and T is a 3x1 vector for translational offsets.

To estimate the calibration matrix TR, we can use a gradient-based search method that minimizes a hand-crafted error function, or we can try to establish the association between the 2D and 3D points and then apply the Perspective-and-Point RANSAC method (PNPRANSAC). We will discuss the later methods. Both methods assume that the object point cloud is separated from the background, which is discussed later in this section.

The relationship between the 2D image data and the 3D lidar data is established below in equations (1-3).

$$I_o(t_0) = M \times TR_c \times P_l(t_o)/z_c(t_0) \tag{1}$$

$$I_o(t_1) = M \times TR_c \times P_l(t_1)/z_c(t_1) \tag{2}$$

$$P_l^*(t_0) = TR_c^{-1} \times M^{-1} \times (z_c(t_0).I_o(t0)) \tag{3}$$

3.2 2D-3D keypoints matching

Hence one key step to calibration is to establish enough 2D-3D point correspondence. Existing calibration methods typically require a specific pattern of known geometry dimension and features (such as a checkerboard) to establish

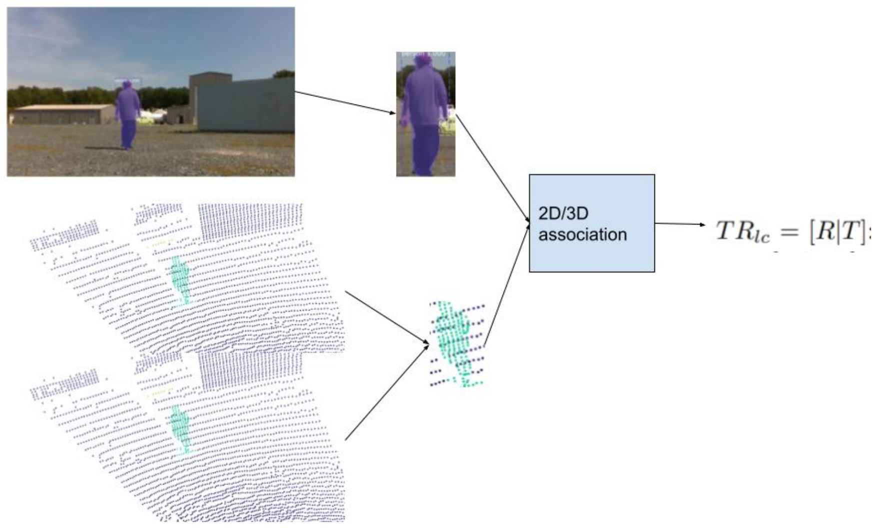

Figure 1: Computing flowchart for TR matrix estimation via 2D-3D bundle adjustment

the correspondence of keypoints in both the 3D frame and 2D image plane. In our case, we would like to use generally shaped objects instead, so an intuitive method is to calculate 2D image features of the target object in image plane, and match those features to those calculated from the projected image points from the 3D point cloud. This requires us to first separate the object point cloud $P_l(t0)$ from the background which will be discussed later in this section. We then calculate the projected object footprint using the initial TR_0

$$I(\hat{t}0) = TR_0(P_L(t0)) \tag{4}$$

If the initial value TR_0 is close enough to the optimal solution, it can be expected that $I(\hat{t}0)$ and the actual $I(t0)$ will been nearly identical subject to an affine transformation. In particular, we can extract keypoints from $I(\hat{t}0)$ and $I(t1)$ and establish their association using a feature extractor such as SIFT. Let $Key(I(t0))$ and $Key(I(\hat{t}1))$ be the respective keypoints extracted. We now have a set of associations between the 2D key points $Key(I(t0))$ and their corresponding 3D counterparts. The least-square error solution TR_{lc} can be readily obtained by applying a PnP solver such as Lu's gradient search method [**?**]

Assuming moderate object movement, it is expected that the delta should remain a very small portion of the object PT.

3.3 Closest 2D point Matching

In this method, instead of using image domain features to calculate a set of key points as 2D-3D corresponding pairs, we iteratively estimate the 2D-3D correspondence for all observed 3D points. We use a similar strategy as the one used in ICP [7] to compute two quantities iteratively: a candidate 3D-2D point correspondence, and the 3D-2D transformation parameters estimated using PnP. At each iteration, we use a *Closest*() operator to predict the subset of the image pixels that are most likely to be the true projection. The resultant pixel set in turn allow a new TR matrix to be computed which move closer to the optimal TR. Intuitively, the reason this might work is exactly the same as the case of ICP algorithm.

```
Input:
    $PC_0, PC_1$: point clouds at two time inst;
  $I_0, I_1$: 2D object mask segmentation ;      $TR_0$: initial calib matrix;
Output:
    $TR_n$: final calib matrix;
Function CaliberationOptimizer
    /* Extract object points from PC$_0, PC_1$                    $*/$
    $(\hat{P_0}, P_1) \leftarrow (PC_0, PC_1)$                              */
    /* /* Calculate 2D Projection of P$_0, P_1$                    $*/$
    $(\hat{I_0}, C_0) \leftarrow$ Projection $(TR_0, P_0)$                      */
    /* /* Build 2D-3D correspondence iteratively                  */
    for $i \in (0, 1, 2, \ldots)$ do
        $I_k \leftarrow$ Closest $(\hat{I_0}, I_{k-1})$                        */
        /* /* Computer the new TR                                 */
        $(TR_{k,i}) \leftarrow$ PnPsolver $(P_0, I_k)$                         */
        /* Apply $(TR_k, P_0))$                                     */
        /* $T_{i+1} \leftarrow \mathrm{T}_i \cup S_{t,i}$
    end
    end
```

Algorithm 1: Iterative Calibration Parameter Search

Here again, we emphasize that $P_{0/1}$ is the estimated 3D point cloud of the target object and not the whole-scene point cloud. This is important to assure the convergence of the algorithm since the *Closest*() operation is only meaningful when the projected pixel set and the observed object pixel set ”mostly” belong to the same object.

3.4 Separation of object point cloud and relative pose estimation

We now discuss the separation of the target object from the background and nearby objects of the two point clouds. The ground plane and static structure

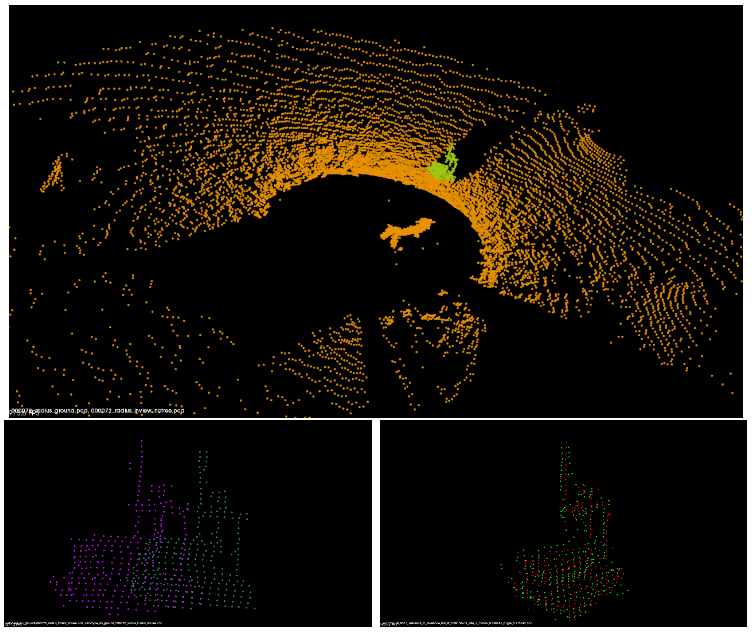

Figure 2: Extraction of the Object point clouds of a Clearpath Jackal robot. (a) first LIDAR point cloud with Jackal object highlighted in green. (b) overlay-ed Jackal object points after a lateral movement. (c) aligned Jackal point clouds after GICP

are processed separately.

We first prune the point cloud with a camera view model such that the remaining points will only contain one moving object and static background. This is a reasonable assumption in calibration scenario. The in-view cut is necessary to remove points corresponding to the human objects, which typically stand behind the robot. The in-view cut allows us to remove 70% of lidar points. It is expected that the delta should remain a very small portion of the object PT. The relative pose estimation method:

(1) GICP to get delta pose of the observing object (2) get $\hat{P}$ from $I(t)$

After the above processing, the size of the point set has about 300 points when the target object (a jackal robot) is about 4 meter in front of the observing robot. Figure 2 shows the extracted point sets at two time instances.

3.5 2D Closest matching

The remaining task is to build a 2D-2D matching between the projected 2D coordinates and the image pixels of the observed object points by the camera, i.e., a mapping function $g(\hat{I}) \rightarrow I_o$ that map a purple points to a green points in figure 2. Noticing that the shape of the projected object points is largely preserved, we opt to use a simple three-parameters linear transformation to approximate $g()$: we first find the centroids for both pixel sets c_0 and c_1, we then calculate the dominant axis at each centroids, $\overrightarrow{x_0}$, x_1, and the associated perpendicular axis $\overrightarrow{x_0}$, x_1. From which the scale factors on the two axis can be calculated as s_0, s_1, as well as a rotational parameter θ. The mapping in 2D image plane is now defined as:

$$(\hat{u_0}, \hat{v_0})^T = [s0, s1]R_\theta(u_1, v_1) \tag{5}$$

$$(u_0^*, v_0^*) \in I_0 \ minimize|(\hat{u_0}, \hat{v_0})^T, I_0)| \tag{6}$$

4 Results

Our basic experiment setup uses a Clearpath Husky robot as the observing robot and a Jackal robot as the calibrating object. The Husky is equipped with an Ouster OS1 32-channel LIDAR sensor to collect the point cloud data. The image data is provided by an Intel Realsense D445 camera. Only the RGB camera data is used in segmentation to generate the 2D image data. The camera is mounted in front of the Ouster sensor. The Husky-Jackal duo is used to evaluate the basic ICPnP algorithm.

Our test scenario has the Jackal robot travel in front of the husky robot across the viewing field. The field is a wooded area so the collected data contain both moving objects, ground plane, and static tree objects. The distance of the Jackal is about 3 meters. We select two LIDAR frames where the object points have considerable overlapping, which would make it more challenging to separate the object points. We set the initial guess of the transformation to be 45 degrees off the ground truth in each of the roll, pitch, and yaw axes. The initial translation error is set to 0.5 meters in all axes. Figure 3 shows the result of the ICPnP algorithm. Figure 3.(a) shows that the initial projection is clearly out of alignment. After one iteration, the ICPnP estimation is already very close to the ground truth. After three iterations, the estimated project is nearly undistinguishable from the ground truth. Table 2 shows the error of the estimated transformation parameters corresponding to the simple closest matching and closest-affine matching. The mean translational errors for both are less than 5 center meters. The rotational error is measured at the three Euler angle axis. The mean rotation error is less than 2 degrees.

Figure 3 shows the effect of cam-lidar calibration in 3D object detection pipeline based on 2D-3D diffusion [1]. With the uncalibrated cam-lidar parameter, about 50% of the object points are mislabeled. After calibration, the object points are labeled correctly.

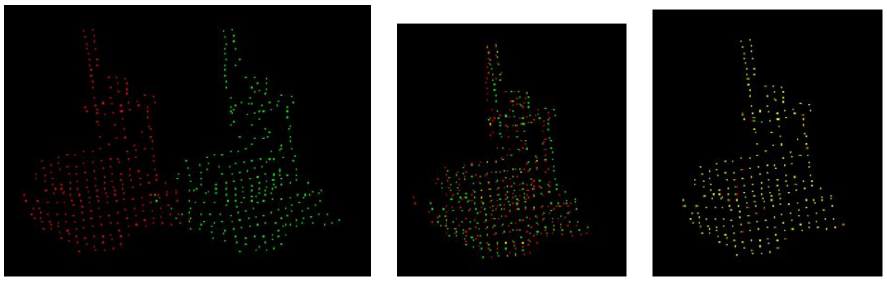

Figure 3: Image plane object points: ground truth (red) and projected (green) using transformation parameters estimated by ICPNP (a) initial guess (b) after one iteration, (c) after 3 iterations.

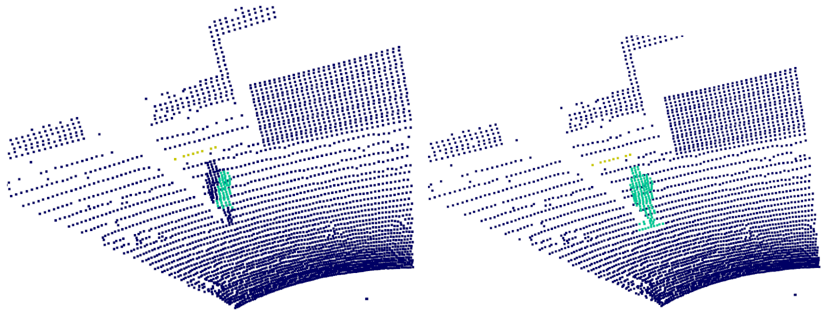

Figure 4: LDLS integration of ICPNP to separate object point clouds with object points in green (a) uncalibrated lidar-camera transformation, (b) ICPNP calibrated

Table 2: ICPNP: Translation and Rotational Error

Method	max TR err	mean TR err	max rot err	mean pix err
closest	0.20	.10	3.25	2.10
closest affine	0.05	0.12	2.32	3.00

5 Conclusions

References

[1] B. H. Wang, W.-L. Chao, Y. Wang, B. Hariharan, K. Q. Weinberger, and M. Campbell, "Ldls: 3-d object segmentation through label diffusion from 2-d images," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2902–2909, 2019.

[2] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Proceedings of Robotics: Science and Systems (RSS '14)*, July 2014.

[3] A. V. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," June 2009.

[4] B. Horn, "Fitting parameterized three-dimensional models to images," in *J. Optical Soc. Am.*, vol. 5, no. 7, 1987, pp. 1127–1135.

[5] D. Lowe, "Fitting parameterized three-dimensional models to images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 441–450, 1991.

[6] L. Quan and Z. Lan, "Linear n-point camera pose determination," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 774–781, 1999.

[7] P. Besl and N. McKay., "A method for registration of 3-d shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[8] D. E. C. S. S. R. C.-Y. F. W. Liu, D. Anguelov and A. C. Berg, "Ssd: Single shot multibox detector."

[9] P. D. K. He, G. Gkioxari and R. Girshick, "Mask r-cnn."

[10] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," 2017.