Improved Techniques for LiDAR-Camera Calibration with ChArUco Boards

EECS C106B/206B - Robotic Manipulation and Interaction

1st Chung, Trinity
EECS
University of California, Berkeley
Berkeley, California
trinityc@berkeley.edu

2nd Li, Alec
EECS
University of California, Berkeley
Berkeley, California
alec.li@berkeley.edu

3rd Lee, Edward
EECS
University of California, Berkeley
Berkeley, California
??@berkeley.edu

4th Im, Justin
EECS
University of California, Berkeley
Berkeley, California
justin.im@berkeley.edu

***Abstract*—This project implements improved methods for LiDAR-Camera calibration for an autonomous car. The goal is to allow a formula SAE car moving at 170+ miles per hour to accurate sense its position. To this end, the car must be able to detect its distance from various objects in its environment, and map objects in the camera to the distances provided by the LiDAR.**

I. Motivation

Researchers have studied LiDAR-camera calibration since the early 2000s. With the rise of self-driving cars and other autonomous systems that must interact with nondeterministic environments, there's an active interest in improving calibration accuracy. The applications go beyond self-driving cars; LiDAR-Camera calibration is useful for determining landing areas for aerial vehicles, mapping and surveying terrain, image segmentation, and even forestry and land management, where it is used for measuring forest density and subsequently used for fire prevention planning. For this project, we are interested in LiDAR-camera calibration specifically as it applies to a self-driving formula SAE car.

II. Related Work

Improvements on LiDAR-camera calibration usually involve either a new kind of calibration target, a new projection algorithm, or both. Cai, Pang, Chen et al. [1] proposes a calibration object that introduces depth in favor of flat calibration boards that contain AR tags. Our approach will be similar - we will use an inverted cube with ChArUco boards on each of the three sides for calibration. Dhall, Chelani, Radhakrishnan et al. [2] proposes a novel pipeline to find the transformation between the LiDAR and the camera using an ArUCo marker setup. For the environment that our implementation will be used in, which is an outdoor track, their method of using a hanging string and motionless board is infeasible. Zhou, Li, and Kaess [3] propose that line-to-line projection should be used over plane projection, and demonstrated that lower error is possible with less pose data. This would be useful for our papplication because recalibration for racing formula SAE cars is common, and one of our goals with this project is speed.

III. Methods

A. Choosing a Calibration Method

In order to use a camera for vision and data processing purposes, we must calibrate the camera, which means obtaining the camera intrinsics and distortion coefficients. Once we calibrate, we don't need to do it again until we modify the camera's optics. The baseline is the current method for LiDAR-camera calibration used by the UC Berkeley autonomous racing team, which involves a single ArUCo tag on a board. In order to improve on the baseline, we introduce a new calibration method that includes depth in the point cloud created by the LiDAR. Our proposed method is an inverted cube with a calibration pattern pasted on each side. Additionally, instead of an ArUCO marker, we will use a ChArUco board. This is due to the fact that we are introducing depth into the LiDAR point cloud, as ChArUco boards profude both ArUCo markers as well as chessboard corner precision, which is important for depth estimation.

B. LiDAR Filtering and Preprocessing

1) Initial Preprocessing: LiDAR data tends to be extremely noisy. As such, we need to take several preprocessing steps before we have suitable data for estimating an affine transformation from LiDAR space to camera space. Outside of the cube, several other elements get caught in the LiDAR's surveillance range and are included in the point cloud. This includes the base that the calibration cube is sitting on, the walls of the room that the LiDAR was run in, the ground,

as well as some background elements. Our first step is to filter out the ground. To do so, we simply filter out points with point cloud normals that are pointing upwards. Through qualitative testing, we found that an angle threshold of 10 degrees worked best. We then filter out the background. Since we know the physical distance from the LiDAR to the cube, all we have to do is set a distance threshold. All points exceeding a certain distance from the LiDAR are excluded from the final point cloud. Doing this is as simple as creating a mask of normals that are less than a certain threshold. Finally, we use clustering to isolate the cube itself. We use Open3D's clustering algorithm, picking an epsilon of 0.2 and choosing a minimum cluster size of 500. We then assume that the largest cluster is the cube, and filter out all other clusters. Finally, we eliminate outliers using Open3D's library.

2) Iterative Closest Point: Although we've isolated the box from the LiDAR point cloud, the planes that make up the box in the point cloud are not perfect planes, i.e. the LiDAR reading includes a lot of noise. To remedy this, we introduce ICP, a point registration algorithm which will allow us to get a perfectly axis aligned box in LiDAR space. Our target point cloud is a generated point cloud of the box, so all we need is a rigid-body transformation. This caveat allows us to use ICP, the conditions for doing so including sufficient closeness between the source and target point clouds in terms of both position and structure. Barebones, ICP is an MMSE optimization problem. FOr each point in the source point cloud, it matches the closest point in the reference point cloud, takes the MMSE, transforms the points, and reassociates and iterates. We ran 100,000 iterations with a threshold of 0.05.

C. Finding Correspondences

Now that we have a sufficiently "clean" point cloud, our goal is to find the optimal affine transformation between the ChArUco point set and the LiDAR point cloud. To do so, we use RANSAC [4]. We first randomly select a subset S of n data points out of N. We estimate the model P from the data points in S. We compute a set of correspondences that are in agreement with the model up to a certain tolerance. If $|S| \geq \epsilon$, then we accept the set as a set of inliers and use it to recompute a better estimate. Otherwise, we repeat from the first step, and stop after max iterations. Once we have an R and t, we project the LiDAR data using our given R and t

$$K \cdot (t + (R \cdot x)) \tag{1}$$

and find the average sum of squared differences between reprojected points and actual ChArUco points.

IV. Experimental Results

V. Discussion

We were able to gain a 3 centimeter reprojection error improvement over the baseline with our new method. The main difficulty we encountered was ICP not finding a suitable transformation and returning a fitness of zero. ICP requires a good initial guess, and we had many failed fits before we were able to find a fit using a beter initial guess. Originally, our design for the calibration object was an outward facing cube, but because of the strength of the lights in the testing room, the ArUCO markers on the top of the cube were almost completely obscured; of course, the CV algorithm was entirely unable to pick them up. As such, we had to redesign our cube and make it an inward facing one. Our work could be improved by using better error herustics, better calibration objects, or filtering by distance using the transformation guess.

References

[1] Huaiyu Cai, Weisong Pang, Xiaodong Chen, Yi Wang, and Haolin Liang. "A Novel Calibration Board and Experiments for 3D LiDAR and Camera Calibration". In: Sensors 20.4 (2020). ISSN: 1424-8220. DOI: 10.3390/s20041130.

[2] Ankit Dhall, Kunal Chelani, Vishnu Radhakrishnan, and K Madhava Krishna. "LiDAR-camera calibration using 3D-3D point correspondences". In: arXiv preprint arXiv:1705.09785 (2017).

[3] Lipu Zhou, Zimo Li, and Michael 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

[4] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, 1981.