MRSD Project Course

Team I – AIce

Autonomous Zamboni Convoy

Individual Lab Report 8

Team

Rathin Shah

Nick Carcione

Yilin Cai

Jiayi Qiu

Kelvin Shen

Author

Kelvin Shen

Oct 13, 2022

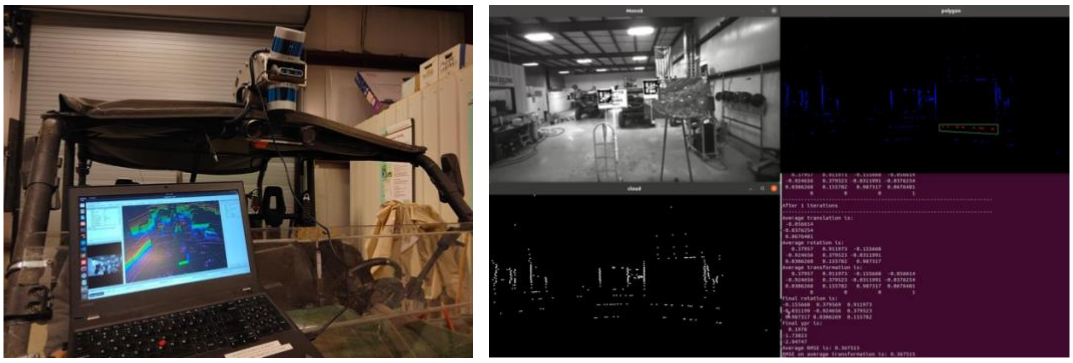
Table of Contents

Individual Progress

During the past two weeks, I have been working on two tasks mainly. One is onboard LiDAR camera calibration, and the other is improving pose estimation results.

LiDAR Camera Calibration

In the last progress review, we were able to run the LiDAR camera calibration package using provided data. However, last time we did not set up the necessary environment nor did we mount the camera onto the appropriate position on ATV. Therefore, our major achievement this time was the correct setup of the calibration and we successfully calculate the extrinsics between camera and LiDAR using the automatic calibration package.



```
  0.37957   0.911973  -0.155668  -0.058614
 -0.924656   0.379523 -0.0311991 -0.0376254
 0.0386268   0.155782   0.987317  0.0676481
         0          0          0          1
---------------------------------------------------------------------
After 1 iterations
---------------------------------------------------------------------
Average translation is:
 -0.058614
-0.0376254
 0.0676481
Average rotation is:
  0.37957   0.911973  -0.155668
 -0.924656   0.379523 -0.0311991
 0.0386268   0.155782   0.987317
Average transformation is:
  0.37957   0.911973  -0.155668  -0.058614
 -0.924656   0.379523 -0.0311991 -0.0376254
 0.0386268   0.155782   0.987317  0.0676481
         0          0          0          1
Final rotation is:
-0.155668  0.379569  0.911973
-0.0311199 -0.924656  0.379523
 0.987317 0.0386269  0.155782
Final ypr is:
 0.1978
-1.73823
-2.94747
Average RMSE is: 0.307515
RMSE on average transformation is: 0.307515
```

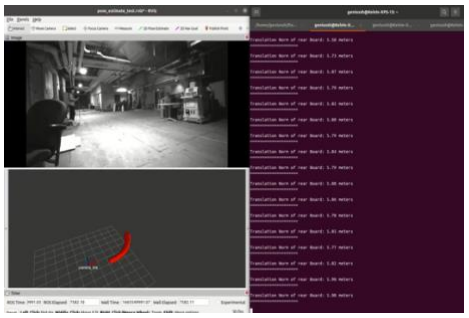
Figure 1. Left: sensor mount position. Right: calibration setup

In Figure 1, we show our setup. The left image illustrates how we mount the RealSense D435i camera relative to the LiDAR. Note that we will only use the bottom LiDAR in our project because the top one is originally installed by the other team to get a dense point cloud representation of the ground plane, which in our case is not useful. Also note that the bottom LiDAR we use is upside down, which implies that we need to provide a correct initial rotation for the calibration package to work properly. In particular, the package works by first rotating the point clouds into the camera frame through the provided initial rotation matrix. Therefore, without providing a reversed relative rotation between camera and LiDAR, the package is not able to converge even given annotations. The setup also involves measuring the board size, the marker size and the margin of the marker. To make sure we can clearly see the edges of the board in the filtered point cloud (bottom left window in the right image of Figure 1), we should place the boards as detached from other objects nearby as possible. Ideally, we should hang them in the air from an invisible string, as the example given in the package documentation. In the filtered point cloud, we don't have a clear top and bottom boundary, which is one thing we must improve during calibrating LiDAR-camera on the Zamboni by rethinking about the setup.

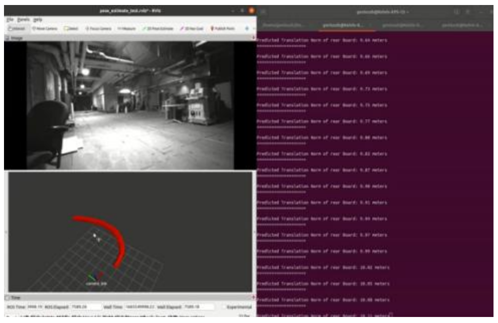
To start iterative LiDAR-camera calibration using 3D-3D point correspondences, we mark each edge of the board. Each board have 4 line segments and need to be marked from leftmost board to the rightmost board. We draw a quadrilateral around the line of points that we believe correspond to an edge of a board in the grayscale image. After annotating all the line segments, the rigid-body transformation between the camera and the LiDAR frames will be calculated with a fixed number of iterations (50 by default). It also calculates the Root Mean Squared Error (RMSE) between the 3D points viewed from the camera and the laser scanner after applying the calculated transformation. Given the transformation from the LiDAR to camera, we can easily project the image onto the point cloud.

Perception Improvement

Last semester we found the leader-following is very unstable because the perception subsystem is not robust against noise. In the last progress review, we discuss options we can improve from the hardware perspective. We ended up switching from the RGB stream to the monochrome stream, which not only consumes significantly less bandwidth but also reaches an FPS as high as 90 in a normal resolution or 300 in a narrow resolution. In this progress review, we improve our software to further make the pose estimation robust.



```
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
Translation Norm of rear Board: [illegible] meters
```

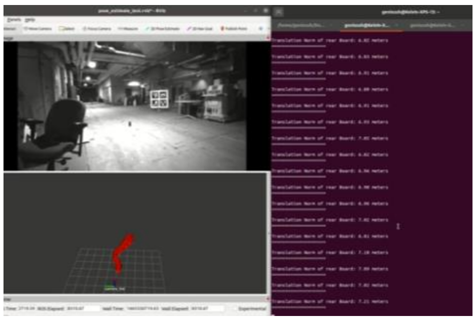


```
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
Predicted Translation Norm of rear Board: [illegible] meters
```

Figure 2. L: the moment beyond which there’s no visible marker. R: the predict step predicts poses

The original pose estimation (purely based on ArUco marker board) is improved in two ways. First, we add a predict step, which functions when there’s no ArUco marker detected. Loss of ArUco usually happens in two cases: (1) motion blur due to aggressive rotations or vibrations of the camera, (2) occlusions that block the marker board from the camera view especially during vehicle steering. During this predict step, we extrapolate linear positions by fitting curves on a fixed window of previous poses with optional sampling or sparcification. The sparsification helps storing a longer history of poses given a fixed storage size, which significantly improves the curve fitting result. Before fitting curves, we also tried the *interp1d* function from SciPy, which did not work well because it interpolates linear/quadratic/cubic lines between every two points, which is very prone to errors when the board is farther from the camera, producing noisier pose estimations. When it comes to rotations, because it's in the SO(3) manifold rather than linear, we extrapolate the estimated rotation by RotationSpline from SciPy, which internally interpolates the quaternions based on timestamps and estimates corresponding rotation given a timestamp beyond the provided timestamps for interpolation. In Figure 2, we show the result of the predict step. The left image marks the moment where the board is no longer visible from the camera beyond this point, so the red odometry drawn in RViz shows the pose estimation results estimated directly from the ArUco board. The right image shows how we are able to predict the pose without any ArUco board by extrapolating the history of poses we have estimated. Note the translation between the board and the camera is around 6 meter, which is exactly the requirement for the longitudinal offset we will maintain between leader and follower vehicles.

When the ArUco marker board is detectable, we modify the original algorithm into an update step (following the same naming as a Kalman Filter). During the update step, we have two options. On one hand, we can fuse the current state (from the previous predict step) with the estimated pose from ArUco. On the other hand, we can directly use the estimated pose from ArUco. The first option suffers from the problem of accumulation of errors even with covariance assigned to the measurements or the predicted state. In our use case, we don't care too much about the smoothness of the leader pose estimate; rather, we look for accurate pose estimation results as many as possible. To this end, we implemented the second option, which was also more straightforward.



```
Translation Norm of rear Board: 6.92 meters
Translation Norm of rear Board: 6.83 meters
Translation Norm of rear Board: 6.91 meters
Translation Norm of rear Board: 6.89 meters
Translation Norm of rear Board: 6.91 meters
Translation Norm of rear Board: 6.93 meters
Translation Norm of rear Board: 7.85 meters
Translation Norm of rear Board: 6.82 meters
Translation Norm of rear Board: 6.94 meters
Translation Norm of rear Board: 6.98 meters
Translation Norm of rear Board: 6.96 meters
Translation Norm of rear Board: 7.02 meters
Translation Norm of rear Board: 6.81 meters
Translation Norm of rear Board: 7.18 meters
Translation Norm of rear Board: 7.09 meters
Translation Norm of rear Board: 7.02 meters
Translation Norm of rear Board: 7.21 meters
```

Figure 3. Stress test under strong vibration of the camera

In Figure 3, we show the stress testing result where one aggressively vibrate the camera during pose estimation. Since we plot the board position relative to the camera (keeping the camera frame as reference), we can see the result is actually very accurate in that it never goes out of its current overall trajectory. Remember in SVD, the pose estimation was very unstable by producing a lot of outliers or noises along the way.

Challenges

The major challenges I have encountered are:

- Camera LiDAR calibration package is not well maintained. A lot of bugs when setting it up on our own sensors were solved by going through all relevant GitHub issues in the community.
- Setting up the environment for camera LiDAR calibration was challenging in Gascola facilities. We didn't have a lot of tools to set up the markers and the boards in an ideal setting as shown in the documentation, which led to noises in our point cloud even after filtering. We utilized the ladders or trolleys we could find in the facility and built an initial setup to test the package on our sensors particularly.
- Choosing the proper extrapolation method for our case was difficult. We spent a lot of time trying to tune the *interp1d* method from SciPy and tested dozens of times per parameter. None of them gave a robust performance because we eventually found it worked by interpolating every two point in the history instead of interpolating globally. Therefore, this problem immediately led us to switching to the curve fitting idea, which worked very well.

Teamwork

- Nick updated Zamboni mount designs and the camera mounts onto the ATV. He brainstormed an initial ATV brake-by-wire design with Rathin and Jim Picard. He was also involved in the localization tests and lateral controller testing at Gascola.
- Rathin brainstormed the ATV brake-by-wire design as well. He was actively involved in testing the localization algorithm and the lateral controller.
- Yilin led the localization tests and waypoint following controller tests on ATV. He worked with Jiayi on the longitudinal controller algorithm improvement. He also provided a lot of help when Kelvin tested and improved the perception algorithm.
- Jiayi integrated the longitudinal controller with other subsystems in simulation and tested the longitudinal controller when taking turns. She wrote waypoint following scripts for lateral controller testing on ATV. She tested the lateral controller with the team on ATV.

Plans

I plan to project the image onto the point cloud and start running YOLOv4 to finish the obstacle avoidance. I also plan to integrate the improved perception algorithm onto the system and tested it on the ATV. In addition, I will decide how many cameras we will eventually need by testing the sufficiency of the current horizontal FoV when the leader vehicle is turning.