Automated Driving Using External Perception

Individual Lab Report - ILR10

November 16, 2023

Team E - Outersense

Author:

Ronit Hire

Team Members:

Atharv Pulapaka
Dhanesh Pamnani
Jash Shah
Shreyas Jha

Carnegie Mellon University

Contents

1 Individual Progress

This PR was supposed to act as a dress rehearsal for our upcoming FVD where we will be demonstrating 2 cars being controlled and driven by 3 infrastructure sensors. Having completed all the unit tests for individual subsystems before, we focused on integration tests. My goal for this PR was to monitor the perception system for extended periods of time, identify bugs and tweak the hyper-parameters related to calibration and tracking.

1.1 Perception

1.1.1 Reliability Testing

The object tracking and pose estimation modules of the perception system were tested during the last PR. Only the object detection and calibration modules remained and I concentrated my efforts on testing them. On the detection side we primarily detect two types of makers - 1) Aruco tag with different IDs which represent our RC cars and 2) Outersense logo which acts as an indicator for static and dynamic obstacles (uncontrolled objects). Aruco detection happens through the standard OpenCV package whereas for the Outersense logo we use Hough transform to identify circles in the image and threshold them based on the radii. For testing these functions, I ran static tests by placing these makers at different locations on the track and validated their bounding boxes. I observed a flickering issue with the outersense logo which is discussed in detail in the challenges section. I also discovered that our extrinsic calibration was suboptimal which led to pose discrepancies in the overlap region of the two cameras.

1.1.2 Extrinsic calibration

Up until now, we used to rely on strategically placed Aruco markers on the ground plane to estimate the camera pose. The arucos which were co-visible in multiple views allowed us to calculate the relative transform between the cameras and we then used these transforms to map an object from the camera frame to the world frame. While testing the detection module we discovered that when an object is placed in the handover(overlap) region, we got varying estimates of the object pose in the world frame. We thus realized that simply relying on a single Aruco is not ideal and we need to refine our camera poses. Since our track is a uniform black surface with very little features, we can't use any correspondence based methods to optimize our camera poses. On the other hand, due to the unique geometric arrangement of the cameras, pixels in the overlap region are an indirect correspondence and we can use them to minimize the reprojection errors. We used a PnP solver from OpenCV to minimize this error and the discrepancy in our pose estimates was significantly reduced ($\leq 7cms$). The output of the optimization can be seen in Fig [1]. The orange and green dots show the misalignment in the uncalibrated case whereas the red and black dots represent the calibrated case.

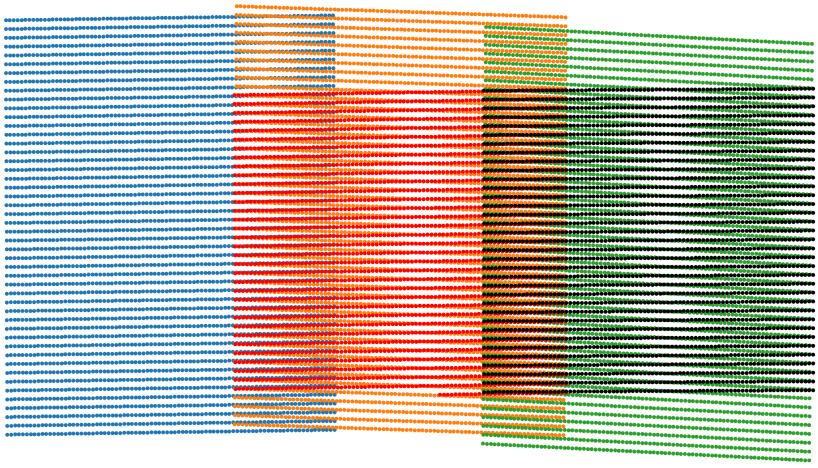

Figure 1: Visualization of image planes before and after calibration

2 Challenges

2.1 Flickering obstacle pose

While the detection of Outersense logo is faster than that of an Aruco marker it is relatively less stable and more prone to false positives. The detection module fails to consistently detect the marker and we observe about ≈ 5 failed detections every 30 frames. This is an area of concern for us because we treat these markers as obstacles and the planning subsystem relies on these detections to update its internal occupancy grid. A missed detection leads to an erroneous map update making it obstacle-free for an instance and resulting in the planner giving a trajectory through the obstacle. To resolve this, we decided to latch onto the pose of the obstacle for 1 second even when it is removed from the track. This prevents a missed detection from adversely affecting the planner and also serves as an additional safety measure.

2.2 Network latency

As the complexity of our system increased, the number of active ROS nodes at any point in time has gone up significantly. The amount of data flowing through our network has increased exponentially and we are observing a noticeable delay between our ROS messages. We have started pruning our system to remove unnecessary subscribers to reduce the bandwidth consumption. We might also have to cut down on some visualizations as they are very data hungry and block the network.

2.3 Erratic cruise control

We use a PID controller in our Active Cruise Control module to update the speed of the follower vehicle when the leading vehicle is a within a distance threshold. This updated speed then serves as an input to the MPC controller. During testing we observed some undesirable behavior from the follower vehicle where it suddenly steers out of the lane, especially on the corners. We are yet to determine the root cause for this and are actively investigating this bug. We suspect a lookahead issue with the MPC and our next step is to tune and test this parameter.

3 Team Work

Along with testing and monitoring the perception system, as a team we also completed the integration tests for Cruise control, Planning and State Estimation.

- **Jash Shah**: Jash worked on completing the planning subsystem by scaling it handle multiple cars. He is also looking into the flickering issue and writing a node to do pose latching. He collaborated with Dhanesh to solve some map related issues.
- **Shreyas Jha**: Shreyas worked on tuning the low-level controller as well as the pose fusion node. He worked closely with me to integrate the heading estimate from perception with the heading estimates from odometry and IMU. On the hardware side he is preparing the electronic components of our backup RC car.
- **Dhanesh Pamnani**: Dhanesh was actively testing and fixing the map related issues in the planning subsystem. He also collaborated with Jash to quickly resolve some of the edge cases in the planning module which might otherwise have blocked our entire system. He is also preparing the chassis for our third RC car.
- **Atharv Pulapaka**: Atharv tested the controls block along with the cruise control module. He identified an issue with asynchronous callbacks which was causing erroneous velocity commands to be sent to the car. He worked closely with Jash and Dhanesh to seamlessly integrate the planner with the controls block.

4 Plans

Moving ahead, my individual goals are:

- Solving the obstacle flickering issue on the perception side.
- Helping other team member test various subsystems.
- Monitoring the calibration pipeline to ensure repeatability for future tests.

As a team our goals are:

- Test and tune the full system for robustness and identify failure points of the new planning and cruise control module.
- Make the test-track presentable for FVD by adding informative labels and markers.
- Incorporate safety behaviours in the cars to deal better with communication loss and high latency cases.