Automated Driving Using External Perception

Individual Lab Report - ILR08

October 12, 2023

Team E - Outersense

Author:

Ronit Hire

Team Members:

Atharv Pulapaka
Dhanesh Pamnani
Jash Shah
Shreyas Jha

Carnegie Mellon University

Contents

1 Individual Progress

The goal of this PR was to run our system at full scale i.e 2 controlled cars driven by 3 infrastructure sensors. This includes listening to a global mission planner for waypoints but excludes local planning for obstacle avoidance. My efforts for this PR were focused on completing the literature review of Multi-Object tracking papers and refactoring the code for efficient pose publishing.

1.1 Perception

1.1.1 Code Refactor

The way our system currently estimates pose of controlled vehicles is that it spawns a new tracking object for each new confident unique detection it receives. The object internally runs optical flow to generate velocity and heading estimates. Moreover, each tracking object is also responsible for publishing its own pose. If tracking for a particular vehicle fails the object is deleted and a new object is again initialized after confident detections. This results in a lot of publishers being created and destroyed which is an quite inefficient. To fix this, I defined a new ROS message type which can store multiple poses and publish them at once instead of each vehicle publishing for itself. This publisher constantly monitors a shared array where poses are populated and updated. After this, we expect reduced load on the compute system.

1.1.2 Multi-Object Tracking

After having narrowed down the set of feasible Multi-Object algorithms for us to implement based on compute, schedule and complexity, I decided focused on "MuSSP: Efficient Min-cost Flow Algorithm for Multi-object Tracking". Min-cost flow has been a widely used paradigm for solving data association problems in multi-object tracking (MOT). MuSSP derives its efficiency from exploiting the special structures and properties of the graphs formulated in MOT. The flow chart of the algorithm and the overall concept can be seen in Fig.[1].
MuSSP consistently beats other implementations of the min-cost flow optimization paradigm like "Follow Me" and "cs2". MuSSP builds on the success of vanilla SSP by using algorithmic improvements like Augment Flow and Residual Graph clipping. After studying their offline code, I am trying to implement an online version of the same which can handle dynamic graph generation.

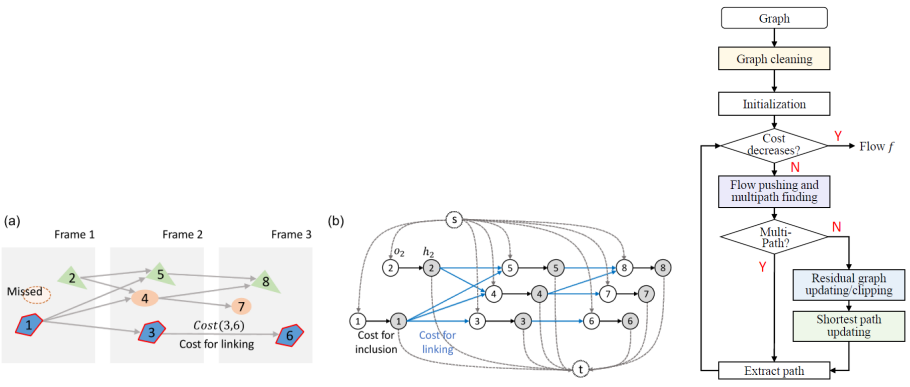

Figure 1: Min-cost flow paradigm and flow chart for MuSSP

2 Challenges

2.1 Hardware failure

Addition of the third infrastructure structure into our system meant that, a new Intel Realsense would have to be calibrated and mounted on the infrastructure unit (a cantilever beam chassis). During the full system test, we noticed that the camera was behaving in an erratic manner and publishing images at a low frame rate of 15 fps. We usually expect a frame rate of 30 fps for the detection and tracking to work properly. Even after firmware updates and ROS parameter changes, the camera maintained a low frame rate. Thus we had to finally replace the camera with our backup piece and that solved the problem. It was challenging to identify this issue because the failure was quite subtle and we initially suspected a software bug which wasted a lot of our time. It also increases our overall system risk as the failure was quite unexpected.

2.2 Clipped steering angles

Our new track layout, figure of 8, demands two full-steering left and right turns from the RC car at the track extremities. We observed that the car behaved differently under left and right steering inputs. Starting with MPC logs, we narrowed down our problem to the low-level VESC controller and found out that it was clipping servo commands in its driver. After properly adjusting the thresholds and recalculating the servo-trim, we were able to achieve equal turning radius under both types of turns.

2.3 Incorrect yaw estimates

Pose estimation in camera-frame for us relies on optical flow to estimate the vehicle velocity and heading. If the vehicle comes to a stop, it reports a zero heading which is inaccurate at times, especially at turns. One of the ways to fix this is to store the history of previous poses and report the last know heading. Alternatively, we can also fuse data from IMU and odometry to rectify the pose. For now, we have chosen the second approach as it integrates easily into our global data association pipeline.

3 Team Work

Along with individual perception tasks, we as a team also made progress on the aspects like Cruise Control, Data Association and Planning.

- **Jash Shah**: After completing the minimal Gazebo simulator, Jash has started developing a planner to avoid the static obstacles on the track. Since his planning subsystem requires inputs in the form of a map and obstacle poses, he is collaborating with Shreyas and me to define data interfaces and ROS messages.
- **Shreyas Jha**: Shreyas sped up the development on the data-association part. He is trying to fuse vehicle odometry, IMU data and pose from perception to increase the accuracy of the pose estimates. While testing on one vehicle, we observed significantly smoother vehicle movement of the RC car.
- **Dhanesh Pamnani**: Dhanesh completed the misson planner to generate waypoints around the track. He is now working on a behavioral planner which implements a state machine to handle intersections. He is also looking at developing a local planner which can handle static obstacles and plan paths around them.
- **Atharv Pulapaka**: Atharv updated his MPC controls block to include an active cruise control module. It is a simple PID controller for adjusting ego vehicle velocity in response to presence of another controlled vehicle in its vicinity.

4 Plans

Moving ahead, my individual goals are:

- Replacing AruCo with more performant tags like STag.
- Experiment with global data association using MuSSP for consistent long term tracking.
- Generating an occupancy grid with poses of all the detected objects in the scene.

As a team our goals are:

- Test the full system for robustness and identify failure points of the new cruise control module.
- Update track configuration to handle static obstacles and accommodate the staging behaviour of the data-association pipeline.
- Integrate the trajectory generated by the local planner with the controls block.