OuterSense

Automated Driving Using External Perception

Individual Lab Report - ILR06

September 14, 2023

Team E - Outersense

Author:

Ronit Hire

Team Members:

Atharv Pulapaka

Dhanesh Pamnani

Jash Shah

Shreyas Jha

THE ROBOTICS INSTITUTE

Carnegie Mellon University

Contents

1 Individual Progress

The goal of this PR was to duplicate our performance from SVD Encore and perform a full scale validation of our entire system. Apart from serving as a hardware test, it would also allow us to revisit individual subsystems and regain our familiarity with the codebase which was lost during the summer. My efforts for this PR were focused on revising our perception pipeline and identifying approaches for scaling the architecture to multi-object detection and tracking.

1.1 Perception

On a high-level, the goal of the perception system is to estimate the pose of the moving vehicle and communicate that information to the controls block. To get these pose estimates we rely on an object detection module and a tracking module.

The object detection module runs at a low frequency of 2 Hz and helps initialize the tracker and the start location of the car. It also acts as a correction mechanism in a sense that it periodically re-initializes the tracker when it starts to drift in consecutive frames.

The tracking module is a high frequency block which runs at about 20 Hz and continuously estimates the change in pose of the vehicle from optical flow. Based on new incoming images and a region of interest defined by object detection, the tracker finds keypoint matches in consecutive frames and generates flow vectors which we use for velocity and position updates.

1.1.1 Multi-Object Detection

Since we had only one controlled vehicle in our scope for SVD, our old object detection module only looked for a single object and ignored other detections. Now, we have extended the functionality to robustly detect all visible vehicles in the image and are working on the capability to independently track each vehicle.

1.1.2 Multi-Object Tracking

To extend the functionality from single object tracking to multi-object tracking we have to tackle the problem of data association. There is also the problem of missed and false detections. To handle these issues, we use tracking by detection paradigm, where we first initialize candidate tracks for all detections and only for objects which have had consistent detection in 5 frames we upgrade them to ”tracked” status and start the optical flow tracker on them. We also monitor missed detections and once detection has failed for more than 7 frames we remove the object from tracked objects set.

We also have to assign tracking IDs maintain them so that the downstream controller can unambiguously send the correct control commands to the correct vehicle. To assign these IDs we use Intersection over Union (IoU) score between detections in consecutive frames as an assignment cost and apply the Hungarian Matching algorithm to minimize this cost. This solves the short term data association problem in the FOV of a single camera. We are still yet to figure out long-term tracking when the object switches from FOV of one camera to another.

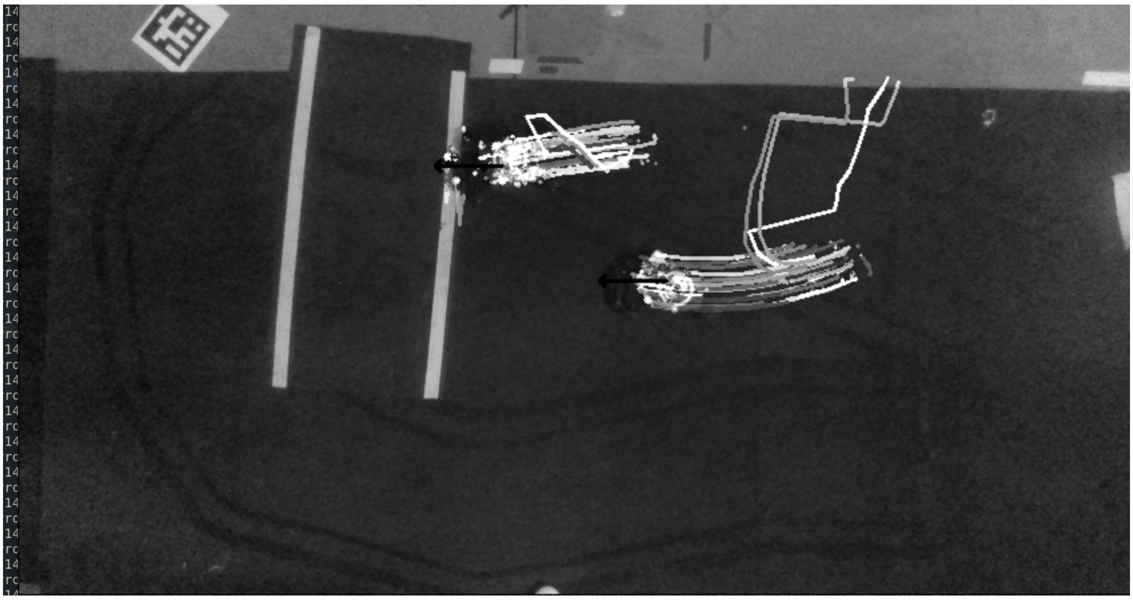

Figure 1: Multi-object tracker in action

2 Challenges

2.1 Embedded Hardware

Our initial plan was to deploy the perception stack as a edge compute node on a Jetson Nano, however, due to architecture incompatibility between the Intel Realsense camera and the Jetson board, we were unable to run our perception stack at the desired FPS. As a substitute, we had to use our laptops as edge devices which blocked development for other members of the team during testing. We tried to resolve the Jetson issue by streaming raw data without any pre-processing but to no avail. We can reliably get raw RGB and depth from the camera at 30 fps, but as soon as we start streaming "aligned-depth" feed we hit a bottleneck. We are trying out some community solutions but if none of them work out we will continue to use laptops as edge devices.

2.2 Dead Realsense

While replicating our SVD test, one of our realsense cameras suddenly died. We tried flashing new firmware but were unable to get it back up. We have sourced a backup camera from one of the other teams and are looking at ways to avoid this scenario in the future.

2.3 Glare on track

Our test site in NSH B-level has strange lighting conditions and one of the light sources is too bright for our cameras. We had a dead-zone in a section of our track due to high glare and have temporarily fixed the issue by covering the light with a piece of transparent foam. We are now able to create some diffuse lighting conditions.

3 Team Work

Along with individual perception tasks, for this PR we also had to run the full integrated system which required collaboration between team members.

- **Jash Shah**: Jash helped re-setup the test-site after the summer break and also contributed to scaling object detection. He will continue supporting different subsystems and independently start working on a Gazebo simulator for testing our planning block.

- **Shreyas Jha**: Shreyas worked on debugging the embedded hardware, trying to setup up data streaming on the Jetson Nano board. He experimented with different Jetpack ubuntu OS to get ROS noetic running reliably.

- **Dhanesh Pamnani**: Apart from setting up the infrastructure units, fixing RC cars and re-calibrating cameras, Dhanesh is designing our new track for FVD. He is also developing a rogue vehicle which will help test the capabilities of our planning subsystem.

- **Atharv Pulapaka**: Atharv has started working on scaling the control subsystem to run multiple instances of MPC. He is exploring multiple approaches to optimize trajectory generation and designing the interfaces for the new planning block.

4 Plans

Moving ahead, my individual goals are:

- Optimizing short term object-tracking to reduce system load.
- Investigating global data association for consistent long term tracking.
- Improving the error handling capabilities of the perception system.

As a team our goals are:

- Setup new track and come up with test plans to understand the limitations of our system.
- Quickly scale different modules to handle multiple objects and test on a smaller track.
- Conduct a thorough literature review on path planning and available ROS packages.