Automated Driving Using External Perception

Individual Lab Report - ILR05
April 6, 2023

Team E - Outersense

Author:
Ronit Hire

Team Members:

Atharv Pulapaka
Dhanesh Pamnani
Jash Shah
Shreyas Jha

Contents

1 Individual Progress

The goals of this PR were perception heavy and I continued with my previous task of architecting a modular software stack so that other team members can easily integrate their functions with the main codebase. Apart from high level design, I worked on getting pose estimates of the moving car from a single infrastructure unit which includes position of the car in world frame, velocity and heading. I also started with a preliminary implementation of data-fusion from multiple infrastructure units.

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 block 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 pose of the vehicle. Based on new incoming images and a region of interest defined by object detection, the tracker finds keypoint matches in consecutive frames and generates a optical flow which we use for velocity and position updates.

1.1.1 Object Tracking

Once a region of interest has been detected by the object detection module, I initialize SIFT features in that bounding box and use them as keypoints for tracking in the next frame. I then apply LK- Pyramidal optical flow to these points and find their closest match in the next frame. The flow equation is given below.

$$I_x u + I_u v + I_t = 0$$

Solving the above equation such that the errors between keypoints in the current frame and the next frame are minimized gives us how much each pixel has moved between consecutive frames.

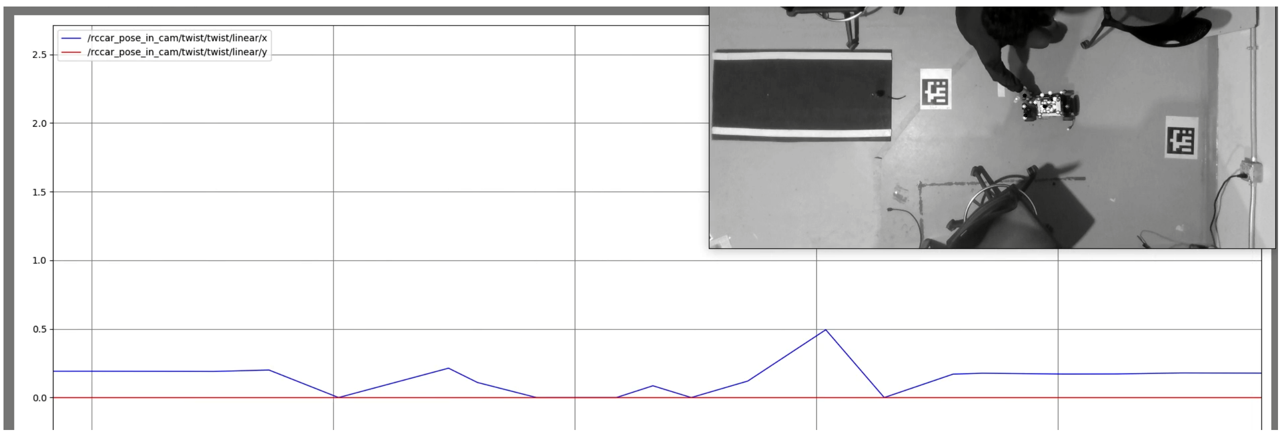

Figure 1: Velocity estimate from perception along with tracker keypoints in the subframe

Since we are using an RGB-D camera, we also have depth information of those keypoints which I couple with the flow vectors to get the distance travelled by each point in the metric scale. Based on this, I then estimate the velocity of each keypoint by dividing the distance travelled with the delta in frame timestamps.
Also, since multiple points are tracked, a slight variation in velocity of different keypoints is observed which I eliminate by taking the median estimate from them. Velocity estimates along with flow vectors are visualized in the figure above.

1.1.2 Data Fusion

In our setup, multiple infrastructure units are observing different areas of the track and we need to stitch them together to get a coherent map. Also, there are instances where the RC car would be present in the field of view of multiple cameras. In that case, the pose of the vehicle from different sources needs to be fused together. I decided to use AruCo markers placed along the track as anchors and use their pose in camera frame to compute relative transforms between the cameras. This will also help in getting the pose of the vehicle in a global co-ordinate frame.

For the global co-ordinate frame, I designate the first aruco marker in the field of view of the first camera as origin and calculate the relative transforms between different cameras and aruco makers by using the 'tf2' package in ROS. A figure showing estimated pose of point in 2 camera views is shown below.

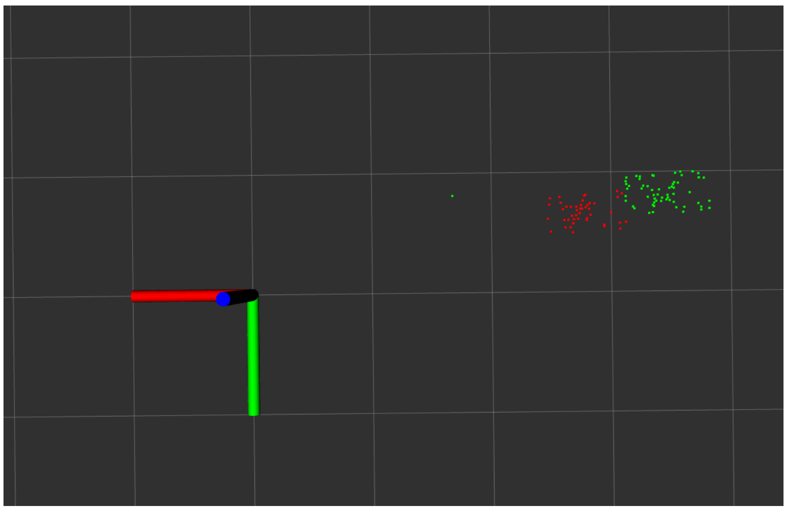

Figure 2: Location of on-vehicle keypoints as seen from different cameras

2 Challenges

2.1 Embedded Hardware

Our initial plan was to deploy the perception stack as a edge compute node on a Jetson Nano and only communicate the vehicle pose to save on network bandwidth and prevent unnecessary delays. However, we soon realized that deploying our current existing system on the Jetson was harder than expected. Our entire software stack is based on ROS Noetic which requires Ubuntu 20.04, whereas all the Jetson boards only support Ubuntu 18.04 L4T by Nvidia which limits them to ROS Melodic and Python 2. Figuring out how to make python 3 packages work for python 2 took a lot of time as most of them especially OpenCV and Pyrealsense had to be compiled from source. Even after these workarounds all of our problems were not addressed and we decided to test our system on Raspberry Pi 4 which has a more powerful CPU. R-Pi 4 is based on the ARM CPU architecture which makes it incompatible with Intel Realsense as there is no out of the box support for that architecture from Intel. We tested some of the solution posted in community forums but none of them allowed us to make our system run at the desired FPS.

2.2 Data Fusion

When the RC car is visible in the field of view of multiple cameras, our estimates of the vehicle pose in the two views differ by a lot which indicates a problem in our camera calibration as well as transformations. Our vehicle, as well as aruco pose estimation is based on depth estimates and we are limited in depth accuracy to a RMS value of 3cm by the realsense hardware specifications. Also, since the aruco markers are placed on the rough ground, the planarity assumption made by aruco pose estimation algorithm is invalid and results in minor errors. We wish to explore generalized PnP solvers for the pose estimation instead of default OpenCV algorithms.

3 Team Work

Along with individual perception tasks, this PR also had some integration tasks between the control and the RC car subsystems.

- **Jash Shah**: Jash worked on the object detection module whose output feeds into the tracking module. He tested different approaches to detect a colored marker on the car like HSV thresholding and template matching. HSV thresholding requires a lot of tuning and he is working on making the template matching more robust and rotation-invariant.

- **Shreyas Jha**: Shreyas integrated a new VESC controller in the RC car which allows for precise low level velocity and steering control. He is able to integrate the commands sent over by the controls block and execute them on the car. He plans to setup NTP on the car to be able to follow new commands based on their timestamps.

- **Dhanesh Pamnani**: Dhanesh worked on the lane detection module which gives an equation for the center-line of the track. This will help the controls block in setting up the desired trajectory by giving waypoints. He is also working on modifying the car chassis to accommodate all the embedded hardware.

- **Atharv Pulapaka**: Atharv worked on integrating the controls block with the RC car. He is able to convert the acceleration output of the MPC controller to a velocity commands which can be executed on the car. He is also tuning the controller parameters to achieve better performance on the real car instead of a simulation environment.

4 Plans

Moving ahead, my individual goals are:

- Reducing noise in the velocity estimates of the perception block.
- Calibrating multiple infrastructure sensors to get a coherent pose.
- Developing a logging module to help in debugging system-wide errors.

As a team our goals are:

- Conducting thorough tests to understand the limitations of our system.
- Fill in any gaps (software and hardware) to increase the robustness of our system.
- Come up with test scenarios which will help us demonstrate the need of a sophisticated control system instead of simple PID controller.