Autonomous Agility Car project report

Introduction

This project involves converting a Traxxas RC car into an autonomous vehicle using ROS2. The car is equipped with a ZED2i stereo camera, an RPLiDAR S1, a Microstrain 3DM-GX5-25 IMU, a Hemisphere GNSS base and rover, and an NVIDIA Jetson TX2i for processing. The final goal is to enable the car to collect data about its environment using these sensors and to navigate autonomously.

Equipment used

Below is the list of sensors and main equipment used in this project:

- Traxxas X-Maxx 8S 4WD Brushless RTR Monster Truck
- Stereolabs ZED2i stereo camera
- NVIDIA Jetson TX2i
- Slamtec RPLIDAR S1
- Lord Microstrain 3DM-GX5-25 IMU
- Hemisphere Phantom 20 GNSS
- Hemisphere R330 GNSS Receiver
- Hemisphere A52 and A45 antennas

ROS2 packages

In this project, ROS2 Humble is used as the main framework for communication between different components. This allows for modularity and ease of integration of various sensors and algorithms. The following ROS2 packages are used/developed in this project:

- `zed-ros2-wrapper`: This package is used to interface with the ZED2i stereo camera. It publishes point cloud data, images, and depth information.
- `rplidar_ros`: This package is used to interface with the RPLiDAR S1. It provides functionalities for LIDAR data acquisition and processing.
- `microstrain_inertial`: This package is used to interface with the Microstrain 3DM-GX5-25 IMU. It publishes IMU data such as orientation, angular velocity, and linear acceleration.
- `hemisphere_gnss`: This is a custom-developed package to interface with the Hemisphere GNSS receivers. It handles communication with the GNSS devices and publishes GNSS GGA data.
- `custom_interface`: This package is developed and used for creating custom messages and services for communication between different nodes.

- `sensing`: This package is used to launch the sensor nodes depending on which sensors are connected and available.
- `localisation`: This package is used to publish localisation data by fusing data from the IMU and GNSS using an Extended Kalman Filter (EKF). The output is local coordinates and is published as an odometry message.
- `bringup`: This package is used to launch all the necessary nodes for the ROS2 system, including sensors and localisation.

Summary of work done

The project started with installing the required software such as the ZED SDK and ROS2 Humble on the NVIDIA Jetson TX2i. However, we ran into compatibility issues with the ROS2 Humble installation on the Jetson TX2i. ROS2 Humble requires Ubuntu 22.04, but the JetPack version compatible with the Jetson TX2i is based on Ubuntu 18.04. We could not use an earlier version of ROS2 as they have reached their end of life and are not supported by the external packages we needed.

To resolve this, we tried to use Docker (the DockerFile can be found in the `assets` directory) to create a container with the `dustynv/ros:humble-ros-base-l4t-r32.7.1` base image, which is compatible with the Jetson TX2i. However, we encountered further issues with building the external packages inside the Docker container. We realised that there was a problem with the CUDA libraries on the Jetson TX2i, which required flashing the Jetson to solve. In order to flash the Jetson, we needed to use a host computer with Ubuntu 18.04 and the NVIDIA SDK Manager. Unfortunately, we did not have access to such a computer, which meant we could not proceed with flashing the Jetson. We researched about using a virtual machine (VM) to run Ubuntu 18.04 on our existing computers, but there were issues with USB passthrough to the VM, which is necessary for flashing the Jetson. Another option is to dual boot Ubuntu 18.04 on our existing computers, but this would require a lot of time and effort to set up and could potentially lead to data loss if not done correctly. After further discussion, we decided against doing this as it would take too much time and effort, and we were likely to face more issues further down the line. The more recently released Jetson Orin Nano Super has Ubuntu 22.04 support, so we decided to order that instead.

We decided to use a computer with Ubuntu 22.04 installed to continue working on the ROS2 packages while waiting for the Jetson Orin Nano Super to arrive. We were able to successfully install and build the required software and ROS2 packages on this computer. We managed to get the ZED2i camera, RPLiDAR S1, and Microstrain 3DM-GX5-25 IMU working and publishing data to ROS2 topics. We also developed the `hemisphere_gnss` package to interface with the Hemisphere GNSS receivers and publish GNSS GGA data to a ROS2 topic.

We tried to make RTK working with the Hemisphere GNSS receivers, with one acting as a base station and the other as a rover. One major challenge we faced was that we were working indoors, but the GNSS receivers require a clear view of the sky to get a good signal. We did not manage to get RTK working, and decided to contact Hemisphere support for help. The solution they provided was required a null modem cable to connect the base and rover directly, which we did not have. For now, we are able to get GNSS data on the receiver connected to the rover, but we will need to get RTK working for better accuracy.

We developed the `localisation` package to combine the IMU and GNSS data using an Extended Kalman Filter (EKF) to provide more accurate localisation. The model uses the IMU

data for orientation and linear acceleration to predict the next state, while the GNSS data is used to correct the position. The output is local coordinates and is published as an odometry message. We tested the EKF using recorded data from the IMU and GNSS, and it seems to be working well.

We also developed the launch packages `sensing` and `bringup` to easily launch the sensor nodes and the entire ROS2 system respectively.

Future work

Once the Jetson Orin Nano Super arrives, it will need to be set up with the required software and ROS2 packages. A `perception` package will need to be developed to process the LiDAR, point cloud, camera, and depth data to detect and classify objects in the environment. A `planning` package will also need to be developed to plan a path for the car to follow based on the localisation and perception data. Finally, a `control` package will need to be developed to control the car's steering, throttle, and brakes to follow the planned path.