INDIVIDUAL LAB REPORT 5

Progress Review - 4

Shubham Garg

TEAM H (PHOENIX)

Teammates:

Akshit, Parv, Steve

April 11, 2019

Individual Contribution:

1. Fire tracking by visual servoing for husky (on Gazebo)
2. Integrate UAV (Iris) and AGV (Husky) simulation on Gazebo
3. Installing Intel Realsense Tracking Camera T265 SDK/drivers on Nvidia TX2

Fire tracking by visual servoing for Husky (on Gazebo):

Setting up the simulation for Husky on ROS Kinetic requires the installation of multiple other dependencies. So, for simplicity, I built the husky package from the source and added LMS1xx, husky, interactive_marker_twist_server, robot_localization and twist_mux ROS packages in the catkin workspace. We had to design a completely new environment (as per MBZ IRC Challenge 3) for the simulation such that the husky and Iris can enter the building and demonstrate fire extinguishing task. The final Gazebo environment can be seen in Figure 1.
I used the move_base ROS package to perform basic autonomous planning and movement on a simulated husky. This ROS package provides an implementation of an action that, given a goal in the world, will attempt to reach it with a mobile base. The move_base node links together a global and local planner to accomplish its global navigation task. Also, it supports any global planner adhering to the BaseGlobalPlanner interface specified in the nav_core package. The move_base node also maintains two cost maps, one for the global planner, and one for a local planner that are used to accomplish navigation tasks.

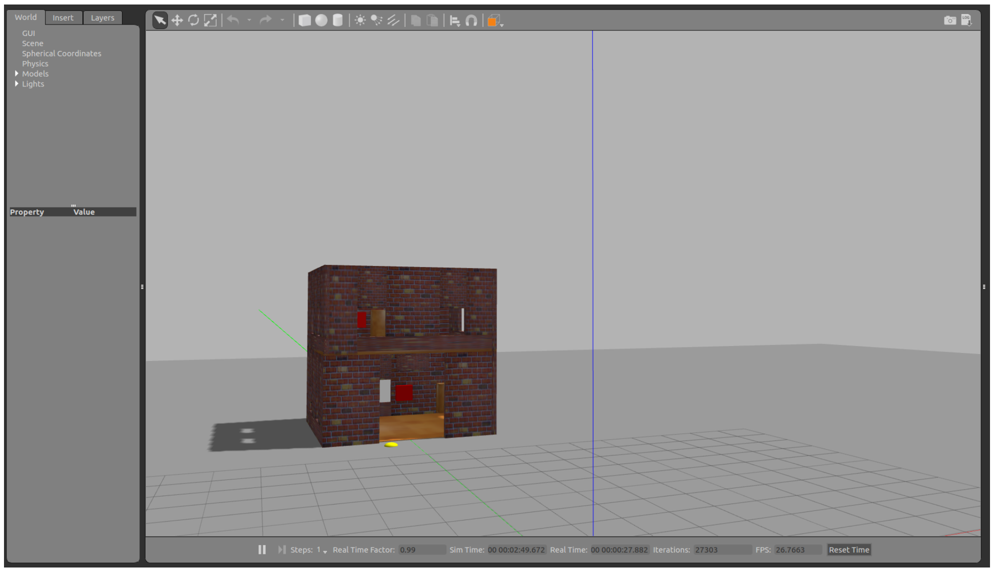

Figure 1: Gazebo Environment for Husky & Iris simulation

Without any absolute localization source on the husky, the position estimate was drifting relative to the world. So, to mitigate this drift, I used the Adaptive Monte Carlo Localization (AMCL) ROS package which fuses on board IMU (odometry estimate) with laser scanner output. This algorithm provides real-time localization for husky (or any other robot). It is a particle filter based probabilistic localization algorithm which estimates the pose of a robot against a known given map. So, the AMCL node uses a generated map of the environment to compare against the incoming laser scans. The output of the laser scans can be seen in Figure 2. For generating the map of the environment, I used gmapping launch file included in the husky_navigation package. Further, I navigated the husky around the environment until it has a good map of the world. Before killing the gmapping, I saved the map with map_server which is used by AMCL ROS node. Also, the Monte Carlo localization algorithm needs an initial pose estimate and without an initial estimate, this approach will not converge to the correct pose.
For simulating visual servoing in the Gazebo, we need to design a perception abstraction framework which can publish the fire location (goal location) in real time. Then that location can be subscribed to send the husky near the fire location. Finally, I created a new launch file which spawns all the required ROS nodes for performing visual servoing. In the current state machine, we are commanding husky to enter the building, go near the fire location and then coming back to the base station.

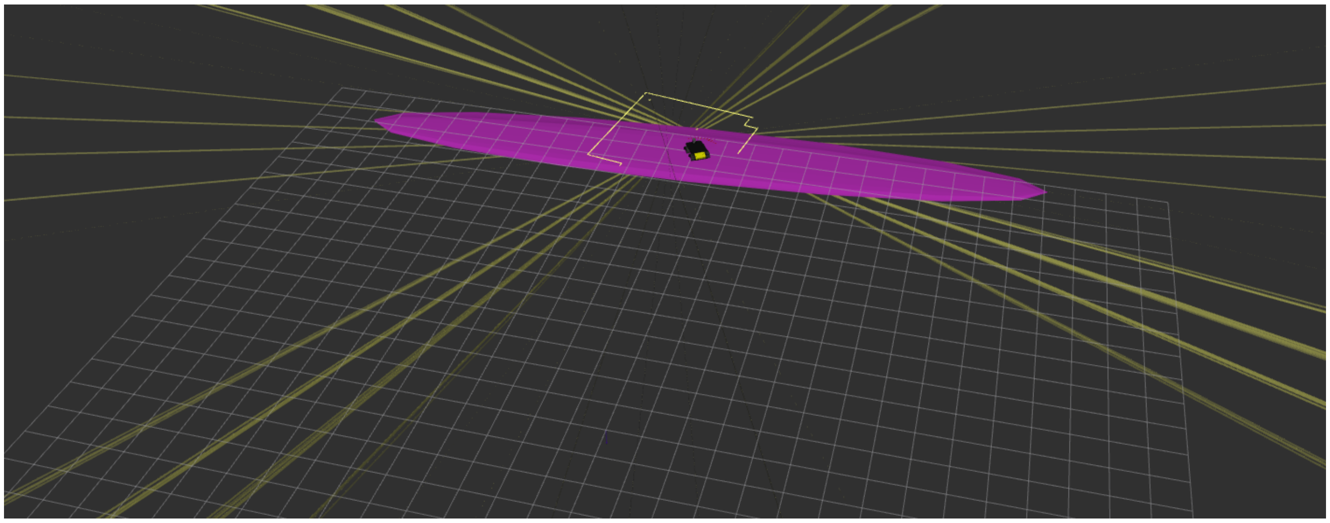
Figure 2: Output of the laser scan on rviz

Integrate UAV (Iris) and AGV (Husky) simulation on Gazebo

ROS uses the MAVROS MAVLink node to communicate with PX4 in the Gazebo simulator. PX4 communicates with the simulator to receive sensor data from the simulated

world and send motor and actuator values. It communicates with the GCS and an Off-board API to send telemetry from the simulated environment and receive commands. So, I recompiled the PX4 firmware for running the software in loop simulation. To run SITL wrapped in ROS the ROS environment needs to be updated, then the required script is launched to start MAVROS on PX4 (publishing IMU sensor data, etc.). And the Gazebo simulation is modified to integrate sensors publishing directly to ROS topics.
Another major task was to integrate the UAV and AGV simulation in the Gazebo. I learned the concept of namespace and tf_prefix, which were essential to ensure that the robots (Husky & Iris) will be able to work independently. So, I created a series of launch files that enabled us in easily adding robots into our Gazebo simulation. While setting up the navigation stack, I ensured that both UAV and AGV operates in its own namespace. To make things more readable I split launch files into few files and created a master launch file which spawns both the robots. Final video demonstration of the Husky and Iris simulation in Gazebo can be found **here**. Both the robots are going towards the building in the Figure 3.

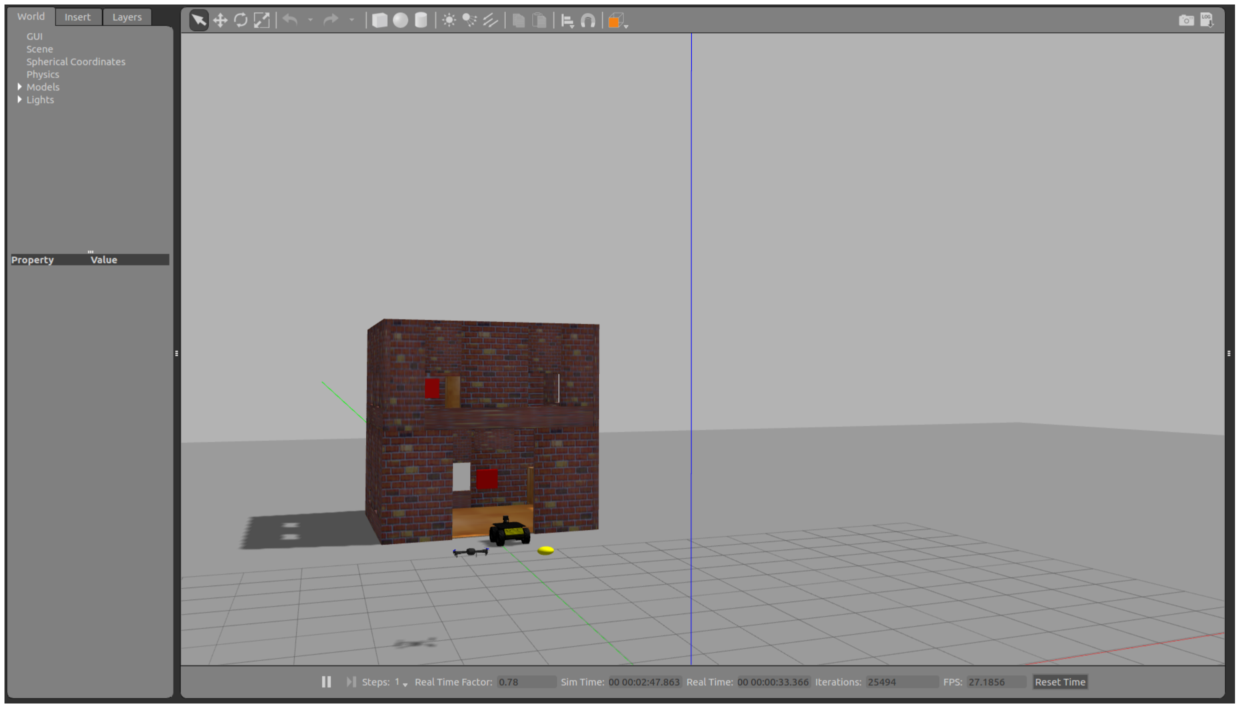

Figure 3: Husky & Iris in action

Installing Intel Realsense Tracking Camera T265 SDK/drivers on Nvidia TX2

Readily available Intel tracking camera drivers work for x86 architectures but there are certain kernel-level modifications required for ARM architecture processor. So, we re-

patched the ARM kernel but then TX2 was not booting up after modifying the kernel. It took me 4 hours to realize that the TX2 is not booting up because the HDMI driver initialization was failing during the bootup. But if the HDMI is connected after TX2 bootup sequence is finished, the display (monitor) was turning on and TX2 was working as expected.
After kernel repatch the D400 real sense camera is working and the tracking camera is getting detected (initialized) but while reading the data from the camera, we were getting some error related to the onboard temperature sensor (real sense). Now, we got another modified kernel which is more specific to the tracking camera which we are going to test next.

Challenges:

Challenges faced in the last two weeks are discussed below:
1. We tweaked multiple ORB SLAM2 parameters to understand its effect on the performance of the SLAM. Since we are still facing scale issues with ORB SLAM2, we are planning to do Indoor navigation using Intel real sense tracking camera. We tried finding the scale of the ORB SLAM2 but it changes with the orientation of the husky.
2. Gazebo simulation added noise in the odometry data and it was causing drift in the husky. This was the case with UAV also and it was drifting towards negative x-axis a lot. So, the simulation was too realistic and now we are planning to change the covariance matrix in simulation for minimizing the noise impact.
3. As mentioned in the previous section, we had to be very careful with the namespace for running UAV and AGV simulation together. We faced multiple issues where UAV was not publishing the MAVROS data while it's running along with the husky simulation.
4. We faced multiple issues while installing Intel realsense tracking camera drivers on Jetson (arm architecture) as it requires kernel re-patching.
5. During testing UR5 arm display controller stopped booting up. We tried debugging the cause but at the end, we reflashed the new Linux image which resolved the issue.

Teamwork

Akshit mainly worked on the setting up ROS framework for UR5 arm, tracking hot water bag, turning on the laser, P controller for the arm and helped **Steve** in establishing the ROS serial link between Jetson and Teensy microcontroller board.
Steve worked on designing perception abstraction layer for the simulation. His major task was to set up the serial interface between the Teensy microcontroller and Jetson TX2 for turning on the laser.

Parv was responsible for tweaking and testing ORB SLAM2 on husky. He also designed the new simulation environment as per the MBZ IRC challenge. We both worked together in writing & testing user defined missions on husky based on the perception abstraction framework.
Shubham worked on tracking fire location by visual servoing in the simulation environment. He also redesigned the environment such that the husky & Iris can enter the building. I worked with **Parv** on writing the state machine for the husky and coordinated **Akshit** for writing the state machine for the UAV.

Future plans

1. We are planning to use Intel realsense tracking camera on both UAV and AGV for the indoor navigation.
2. Our major task is to test user-defined missions in the real world UAV and AGV. This work will be shared among all our team members.
3. **Shubham** will be responsible for bringing up the power distribution for the husky.
4. **Akshit** and **Shubham** will work on creating a new mount for the attaching payload on the UAV. Also, we will work together for testing UAV with 1.5 kg payload.
5. **Parv** and **Steve** will be primarily responsible for integrating multiple software/hardware blocks on husky.