INDIVIDUAL LAB REPORT 10

Progress Review - 11

Shubham Garg

TEAM H (PHOENIX)

Teammates:

Akshit, Parv, Steve

November 6, 2019

Individual Contribution:

1. Longer missions on Husky to test state estimation
2. Mounts and electronics for actuating water pump on Husky
3. Autonomous entering through opening for Husky
4. Setup for running ROS on multiple computers

Longer missions on Husky to test state estimation:

For the last couple of weeks, we were facing issues in fusing the IMU sensor with wheel odometry using a robot localization ROS package. After reading online forums, I realized that the covariance matrices for the IMU need to be fine-tuned. Also, when measuring one pose variable with two sensors, a situation can arise in which both sensors under-report their covariances. This can lead to the filter rapidly jumping back and forth between each measurement as they arrive. In these cases, it often makes sense to only use an absolute pose from one sensor and its velocity from others. When the differential mode is enabled, all absolute pose data is converted to velocity data by differentiating the absolute pose measurements. These velocities are then integrated as usual. So, in our case differential mode is set got IMU and tracking camera while it is set false for the wheel odometry. Only accelerations (in x and y), linear velocities & angular velocities are fused from IMU and angular velocity from the T265 tracking camera. As shown in Figure 1, we drove Husky for around 120m and the accumulated localization error (drift) was around 0.35m.

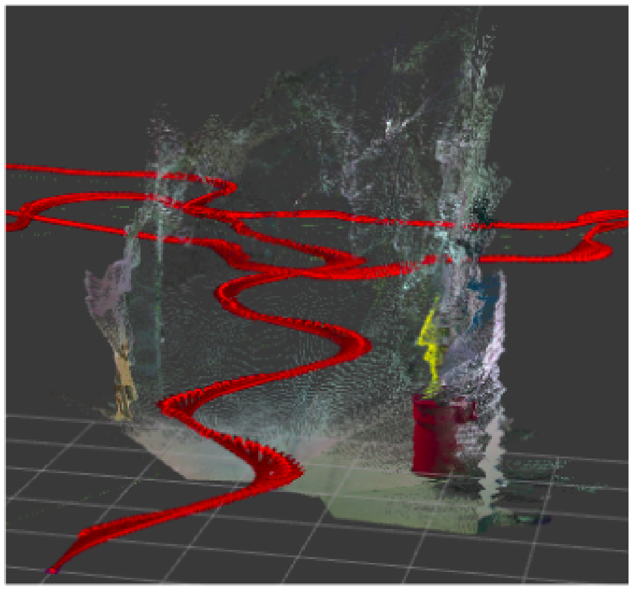


Figure 1: Multiple rounds of NSH level B (Red arrows shows the trajectory of Husky)

Mounts and electronics for actuating water pump on Husky:

Earlier we had three separate mounts for connecting water tank, camera to the UR5 arm. But now we unified all of them in a single mount as shown in Figure 2.

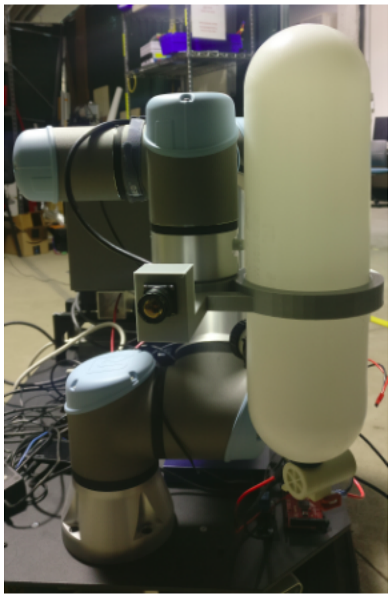
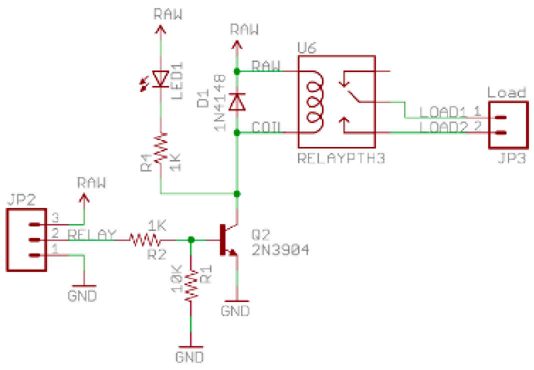
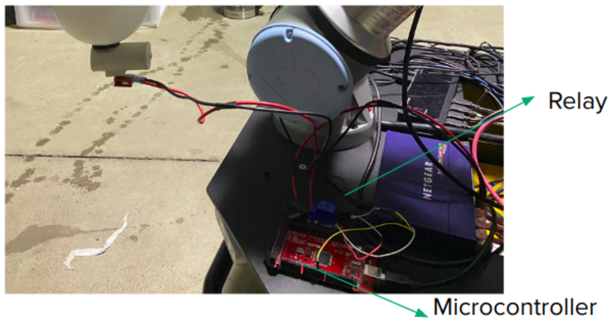


Figure 2: Water tank and thermal camera mount for UR5 arm

First, we were powering our water pump from the UR5 arm's control box but it was not able to source more than 500mA current while our pump requires around 2.5A. So, we are now powering it from an external battery of 11.1V. Figure 3 demonstrates how a relay is to be used to activate the water pump. Power wire (+12V) is connected to the COM (common) and NO (normally open) pin is connected to the device. So, when the relay turns on, power wire (+12V) gets shorted with NO pin and the circuit gets completed to turn on the water pump.





Relay schematic

Figure 3: Electronics for actuating water pump on Husky

We are using rosserial ROS package to communicate with the Arduino as it provides a ROS communication protocol that works over Arduino's UART. It allows Arduino to be a full-fledged ROS node that can directly publish and subscribe to ROS messages, publish TF transforms, and get the ROS system time. Since all the thermal processing is done on the Nvidia Jetson, it publishes the command to activate/deactivate the water pump. This command is subscribed by the Arduino which in turn actuates the water pump accordingly.

Autonomous entering through opening for Husky:

Point cloud data from all the three front cameras are converted into the laser scan. This laser scan gives us reading from $0°$ to $180°$ in a resolution of $0.5°$. Then this information is used to find the rising edge and falling edge in the laser scan. Whenever there is a rising edge, it is marked as the starting point of the opening and falling edge as the endpoint of the opening. This (r,θ) pair is converted into (x,y) coordinates to get the centroid of the opening which will be the goal position for UGV.

Setup for running ROS on multiple computers:

All the thermal image processing is done on the Nvidia Jetson. So, we need to ensure that the Jetson can communicate with the NUC for the seamless actuation of the water pump. Fortunately, ROS is designed with distributed computing in mind. A well-written node makes no assumptions about where in the network it runs, allowing computation to be relocated at run-time to match the available resources.
In ROS, we can only have one master and all other nodes must be configured to use the same master, via ROS_MASTER_URI. There must be complete, bi-directional connectivity between all pairs of machines, on all ports. In our case, Jetson and NUC are connected via an ethernet cable to a switch to ensure that they are in the same network. Both NUC and Jetson advertise itself by a name so that they can resolve their namespace.

Challenges:

Some of the challenges faced in the last two weeks are discussed below:
1. Understanding the behavior tree framework was time-consuming.
2. Integrating various nodes by making them compatible with the behavior tree.
3. During testing, we realized that some nodes like Opening detection were getting very long to respond and the behavior timer was getting expired. So, we added an asynchronous multi-threaded callback to deal with the time-consuming nodes.
4. To fuse the IMU with wheel odometry, we had to try multiple combinations of different

states and fine tuning IMU fusion parameters.
5. We faced multiple crashes while testing missions due to uncontrollable circumstances.
6. We resolved multiple leakage issues in the extinguisher and we had to be extra careful as there is a lot of electronics around it.

Teamwork

We all helped each other in debugging various issues and brainstorming different approaches and algorithms.
Shubham worked on writing the code for opening detection for Husky and **Steve** helped him in designing the algorithm. **Akshit** & **Parv** helped him in debugging various parts of the code.
Akshit & **Parv** were mainly responsible for writing and testing the Behavior tree on UAV and **Akshit** wrote a script to test some missions on the UAV.
Shubham worked on fine-tuning, debugging & improving the state estimation on Husky by adding IMU.
Steve helped all of us in setting up the logistics and arena for testing UAV & AGV.
Shubham wrote the script to test sensor fusion algorithm for longer mission.
Akshit & **Shubham** worked on designing electronics for actuating the water pump on UAV and AGV and wrote script to test the communication between NUC and Jetson. **Akshit** tested the last semester code for fire detection and tracking code on both UGV and AGV.

Future plans

1. **Akshit** and **Shubham** will be working on fixing the power issue on UAV for controlling water pump
2. **Akshit** & **Parv** will be responsible for testing the complete missions on the UAV.
3. **Shubham** & **Steve** will work on writing and testing missions for the Husky.
4. **Shubham** & **Akshit** will be work on integrating the code for establishing communication between UAV and AGV into the framework.