MRSD Project 1

Individual Lab Report #05

Parv Parkhiya

April 10th, 2019

Team H:

PhoeniX

Teammates:

Shubham Garg
Akshit Gandhi
Zhihao (Steve) Zhu

Individual Progress

I started working on the simulation. I was given the task to set up a base simulation on which other team members can implement their individual task. First, I created a simple two-floor building in the gazebo as seen in figure 1. Red block in figure 1 represents a hot water bag (simulated fire).

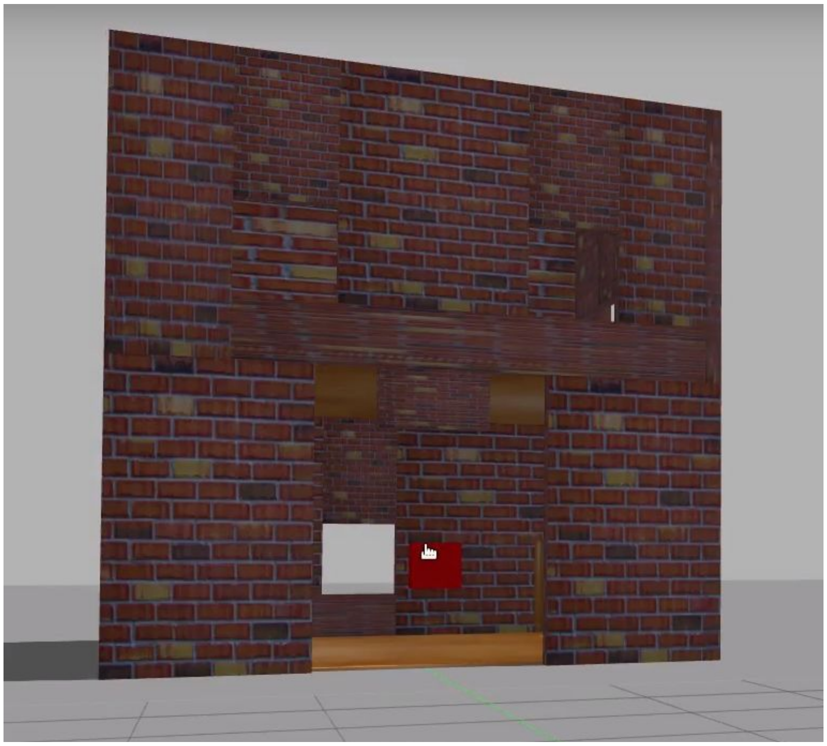
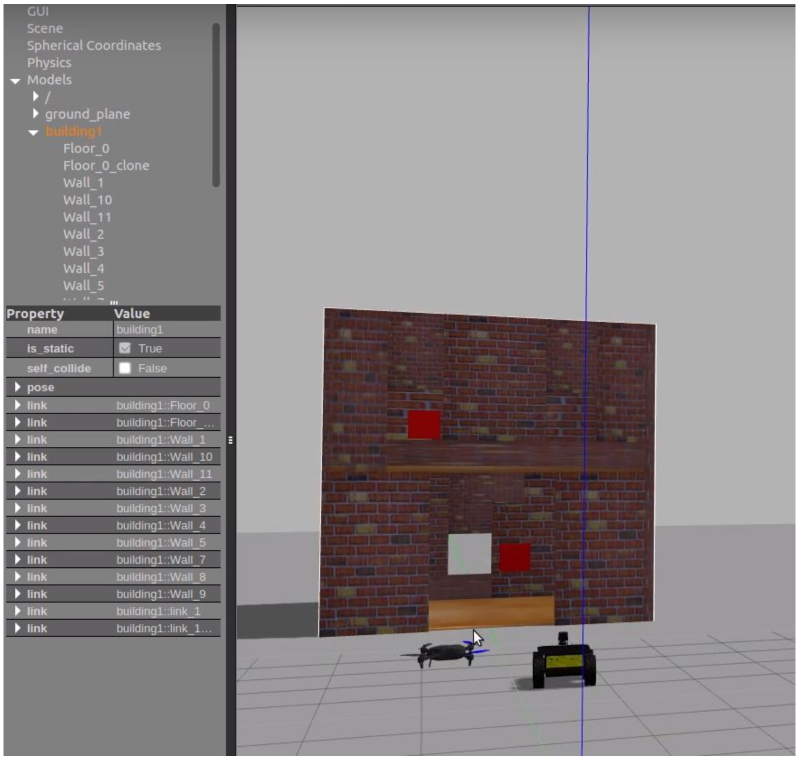


Figure 1: Building in simulation

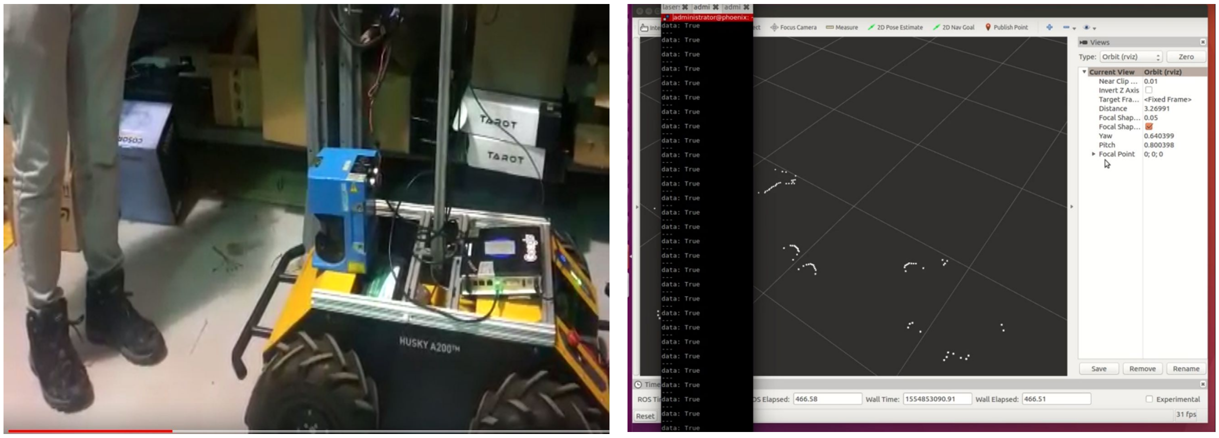
Next task was to spawn AGV and UAV in the scene with their simulated controller. This task turned out to be really difficult. Husky and Hexacopter both have very elaborate simulator routine. When I was spawning both in the same custom world, the simulation was crashing due to internal conflicts. I would discuss more details about the issue in the challenges section. Once that was figured out, I shared the simulation files and tips regarding installing dependencies for the simulation to other team members. Various team members wrote a different part of the simulation. I specifically worked on husky controller part along with Shubham and created a simple state machine with state representing tasks like entering the building, orienting towards the fire and returning back to base.



Property	Value
name	building1
is_static	☑ True
self_collide	□ False
pose	
link	building1::Floor_0
link	building1::Floor_...
link	building1::Wall_1
link	building1::Wall_10
link	building1::Wall_11
link	building1::Wall_2
link	building1::Wall_3
link	building1::Wall_4
link	building1::Wall_5
link	building1::Wall_7
link	building1::Wall_8
link	building1::Wall_9
link	building1::link_1
link	building1::link_1...

Figure 2: UAV and AGV moving towards building

Another major task was to integrate LIDAR with our system. The process went relatively smooth in integrating LIDAR. We had briefly interfaced with LIDAR on Husky, a couple of months back. I had a relook at documentation regarding how to set up and run the LIDAR node. After that, I modified our ROS motion server framework to make sure that whatever forward move command sent to husky can be forfeited if there is an obstacle in front of it. An example can be seen in figure 2 where the image in right shows human legs detected as an obstacle in front of the husky causing husky to stop the forward motion.



```
administrator@phoenix:
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
---
data: True
```

Type: Orbit (rviz) Zero

Current View	Orbit (rviz)
Near Clip ...	0.01
Invert Z Axis	□
Target Fra...	<Fixed Frame>
Distance	3.26991
Focal Shap...	0.05
Focal Shap...	☑
Yaw	0.640399
Pitch	0.800398
Focal Point	0; 0; 0

Time

ROS Ti... OS Elapsed: 466.58 Wall Time: 1554853090.91 Wall Elapsed: 466.51 □ Experimental

31 fps

Figure 3: Husky stopping motion detecting human legs as obstacles

Finally, the most challenging part was to fix problems with ORB-SLAM2 which we faced during previous experiments regarding fewer features on the plain open environment and ORB-SLAM2 not working with the 720p video stream. I was able to resolve both issues by tweaking ORB-SLAM2 hyperparameters. However, when we tried a simple mission purely using ORB-SLAM2, there were some unexpected problems with changing scale while rotating causing large drifts while exploring an unknown environment. Ultimately, the team decided to move away from ORB-SLAM2 and use Intel tracking camera.

Challenges

The most time-consuming challenge was running Husky and PX4 simulation together. Both the simulation are elaborate and having multiple nested launch files embedded deep inside. Because of some internal conflict, gazebo ROS controller server used to crash. After searching for the solution online, I found that standard solution to deal with such internal conflicts is to use ROS namespace method. Which I tried but it didn't work because the nested code was ignoring namespace settings. Finally, I had to dig deep inside various sub launch files to modify the code to allow for ROS namespace system to take effect. The problem was with ROS namespace settings not passing recursively with sub launch files. Using the method above, I was able to get the husky controller working in the simulation with UAV. However, for UAV, there were still issues. For UAV simulation, we were using AIR lab’s custom simulation which was again not compatible for multiple systems running together. Considering time constraints, I added IRIS quadcopter instead of hexacopter to test our high-level state machine code.

Teamwork

Since I had set up the base simulation. I had to collaborate with other team members to make sure they were able to run the simulation. Akshit and Shubham also faced issue while installing dependencies and I was able to help them with common errors. Shubham integrated servoing towards waypoint for which additional libraries were required. Shubham helped me with setting up those on my system. Another challenging part was with perception abstraction in the simulation. Perception abstraction was assigned to Steve. Steve and I discussed what kind of abstractions make sense for our simulation and how much realistic the simulation can be made given the time constraints and Steve implemented the module.

Future Plans

Testing with Intel tracking camera is the next step. We will have to integrate tracking camera with our framework and also figure out a way to mount the camera. We would also have to perform multiple UAV flight test and figure out the way to attach the 1.5 kg payload to UAV. PCB needs to be populated and tested before its use in Husky. UR5e arm's gain needs to be tuned for smooth motion. Finally, we have to bring everything together and do mission tests as per our SVD requirements on both UAV and AGV.