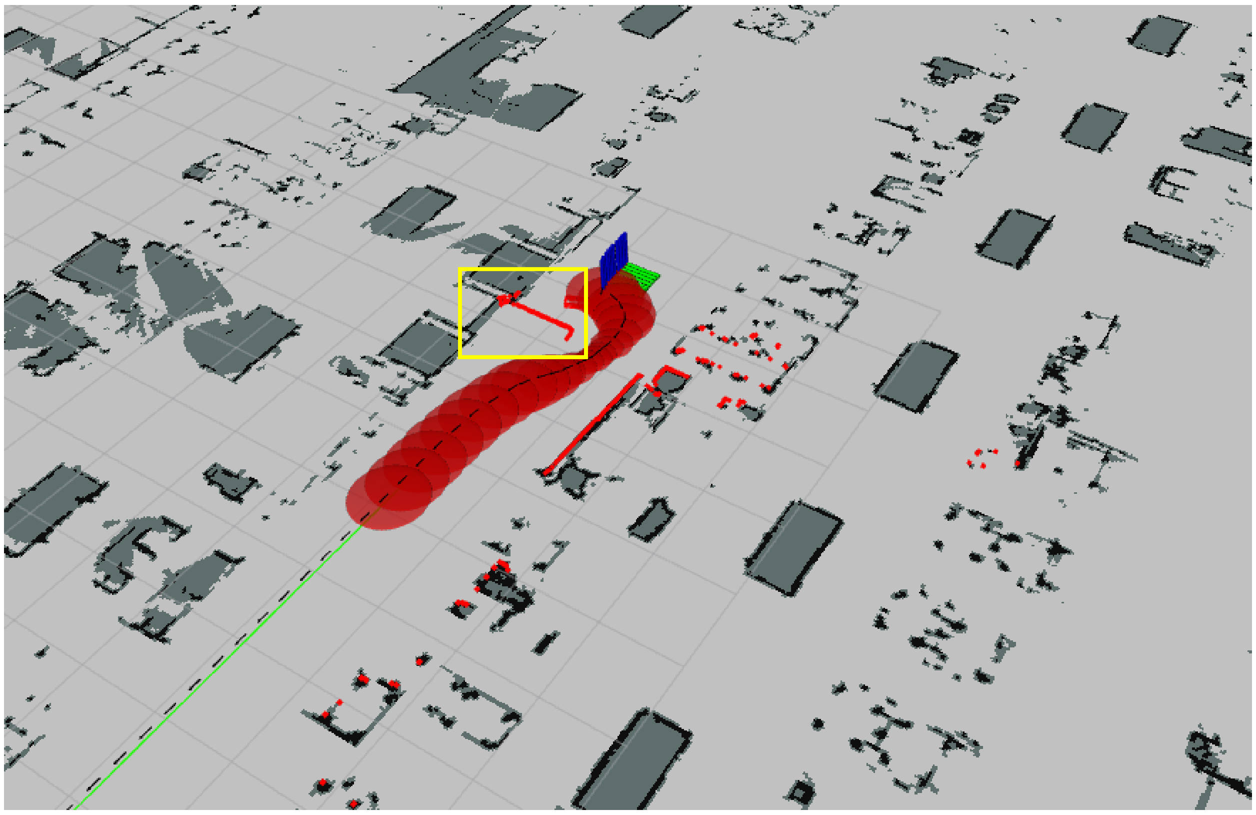
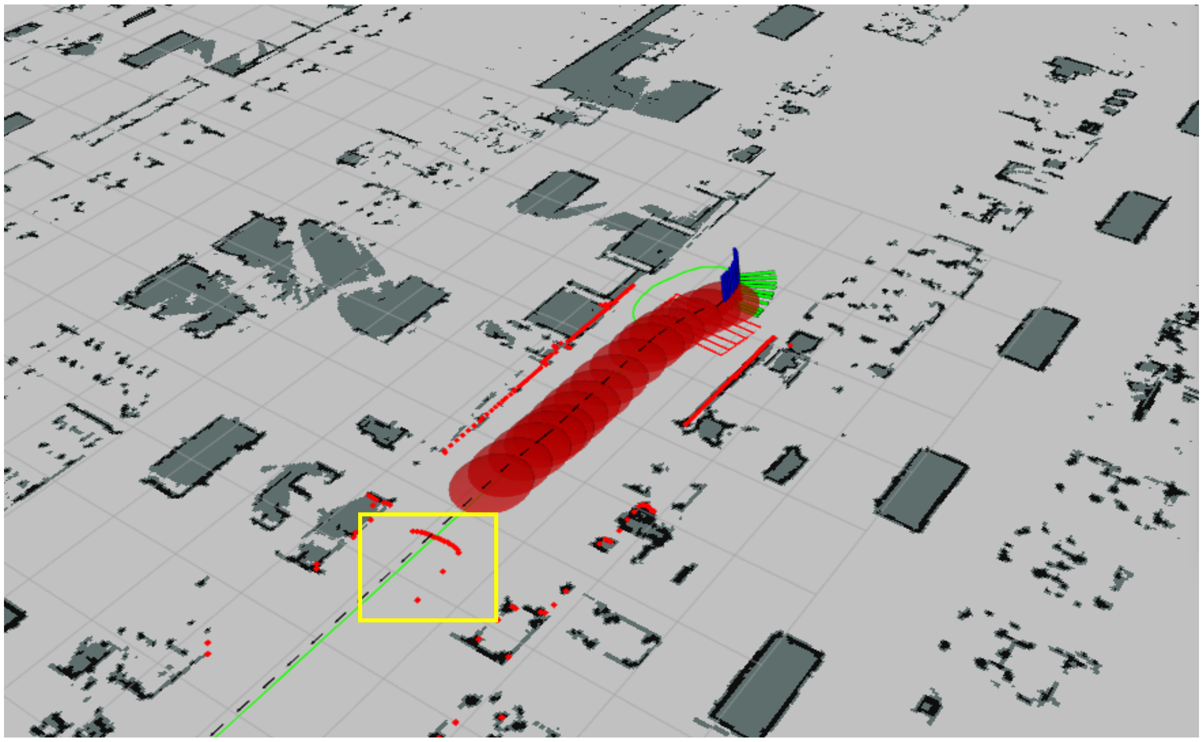
Task 1 Error Analyzing – Verdict and Solution

The timeline given by the customer was from 11:20 to 11:30. The rosbag was thoroughly played and investigated the rostopics that it offers. Out of many topics**, /move_base topics, /scan, /map, /odom, /obstacle_collision_filter/obstacle_zone and /long_term_slam/particles** were useful for the analysis of the rosbag on RViz tool. Some of the other crucial topics that were investigated to understand the problem were: **/stop_signal, /nav_controller/error_code and /long_term_slam/error_code.**

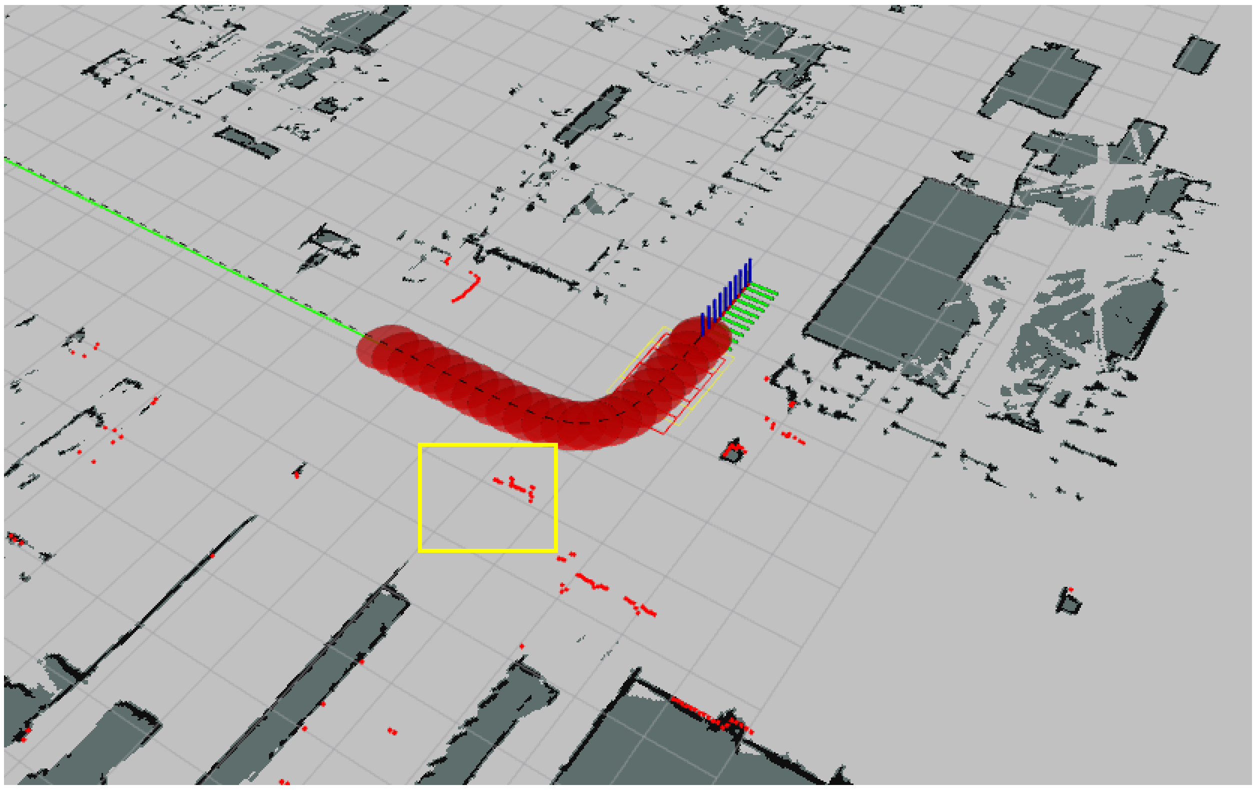
1. The bag clearly starts with a normal functioning of the AMR as shown below. We can see the odometry, laser scan, planned path, ebubbles and the obstacle avoidance region. At this moment, the AMR has stopped due to another AMR that is present in front of it. This can be clearly seen by the laser scan data as marked in the yellow square.



Once the obstructing robot moves forward, the AMR continues to follow its planned path. As shown in the figure below.



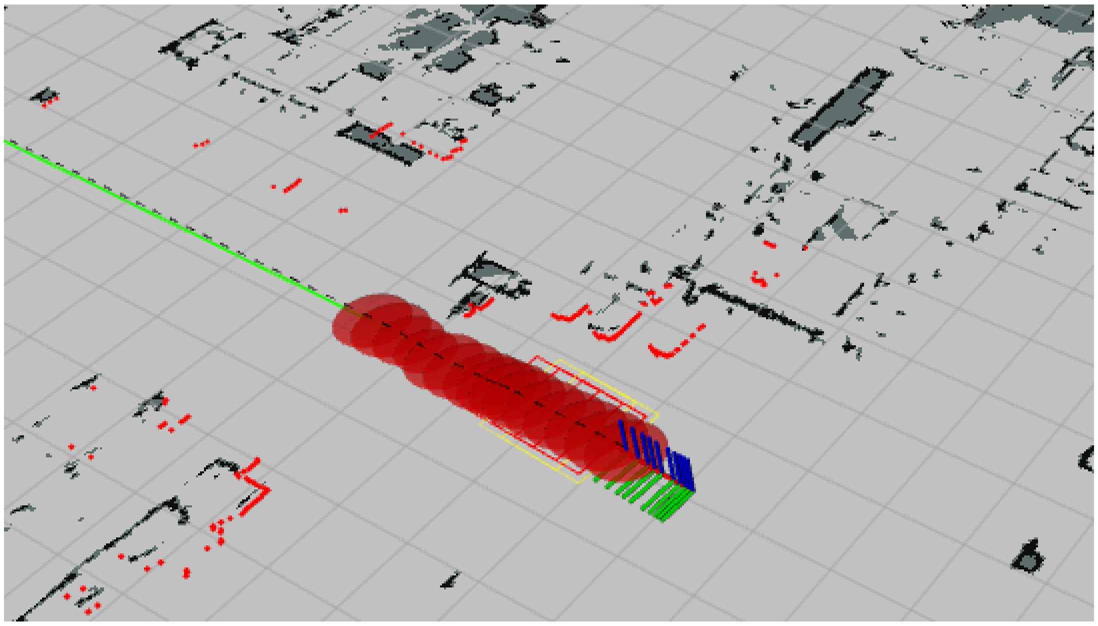
2. Further through the alley, the AMR continues to follow the planned path and ready to take a turn as shown below.



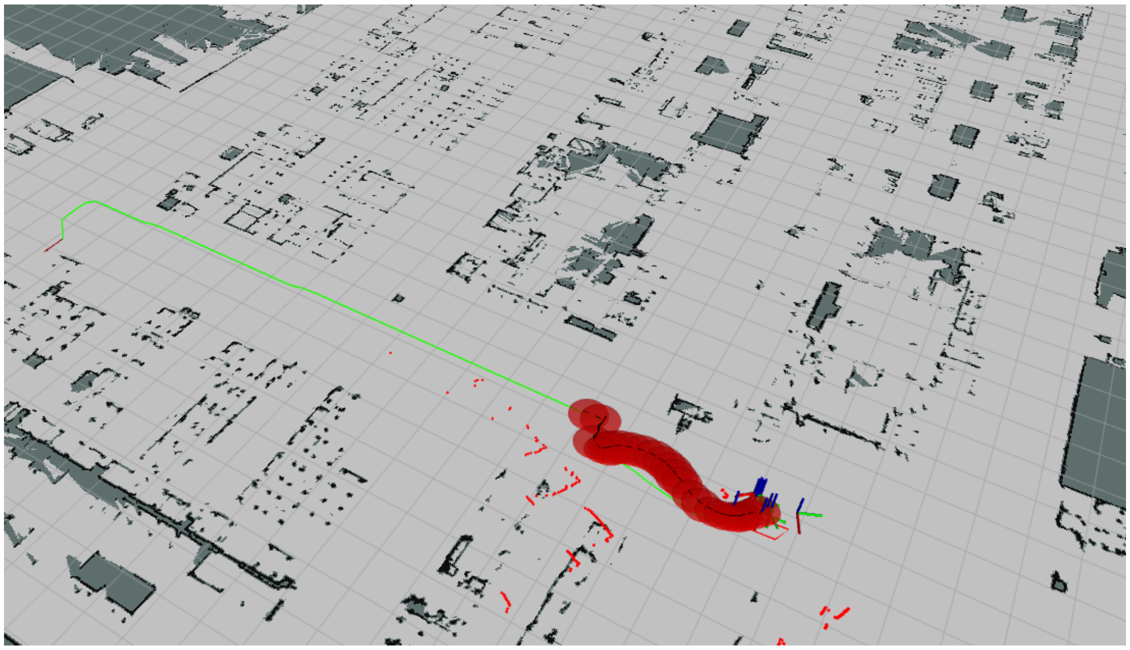
Even in the figure above, we can clearly see another robot moving in the vicinity of the AMR's laser scan.

3. The customer gives information that the robot stops abruptly in front of a dolly. That observation is recorded as shown in the figure below.

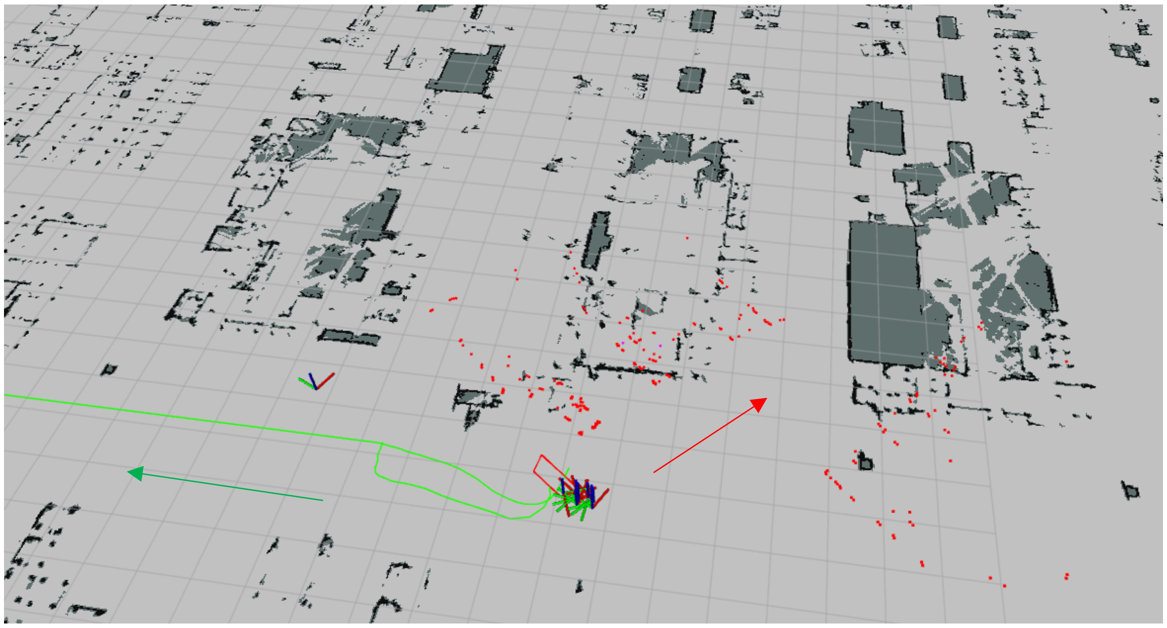
   Just before the incident:



   AMR stops abruptly:



The odom topic visualization rotates around that position and unable to fix its state at one pose.



Finally, the odom topic visualization as shown is stuck at this pose. It was also observed that the recovery planner of the AMR is triggered and replan the path to the goal.

Reason behind this:

First verdict looking at the behavior is the ***Odom jump***. This happens due to false localization and the AMR at this incident is unable to localize itself in the environment. This can be proved by looking at the scan data in the above image. The actual pathway to the left (denoted by green arrow) is misaligned (denoted by red arrow) and the scan data parallel to the red arrow marks the proof of misalignment. This incident is a localization error due to which odom jump occurred and an external routine call the /stop_signal of the AMR to be TRUE resulting in the stoppage of the AMR.

To verify this, the topics **/stop_signal, /nav_controller/error_code and /long_term_slam/error_code** was continuously monitored to get any insight possible error. At this incident, there are error codes that were published on the error_code topics and simultaneously the stop_signal was TRUE. The codes were mapped to the internal codes of the NODE robotics error_code.yaml file. It was evident from this lookup that it was indeed an odom jump or localization issue. At some point when the robot is stuck, the system tries to do a localization recovery and replan the path the destination. But fails to do so in that attempt.

Also, this kind of issue happens when the AMR is unaware of the new environment or simply “Unknown Obstacles” or maps are not updated regularly.

(Reason of Doubt:)

It is also possible that the dolly's ground clearance is much higher than that of the AMR's scanning plane. Normally, AMR's is integrated with local mappers and local planners to tackle any dynamic obstacles or the change in the environment that is not recorded when mapped. Even though if we assume that the dolly's ground clearance is lower than the scanning plane of the AMR and is detected in the scan data, it can be inferred due to some changes in the environment, the AMR is unable to localize itself. Due to which an external routine stops the AMR at the same location.

Possible solution:

1. Updating the current map with this region with newly scanned data.
2. Re-localize the AMR manually, lifting the stop_signal and triggering the local planner with the old goal should allow the AMR to replan and reach its destination.

Task 2: Automatic Error Detection

Using the 3 topics that provides error codes and the stop signal:

- /long_term_slam/error_code
- /nav_controller/error_code
- /stop_signal

A rospackage named "rosbag_analysis_task" was developed to take a rosbag as input and give out a report log file as the output with suitable error details and possible reason for the error. It was developed on ROS Noetic with Rosbag API using C++.

Source of the package on GitHub repository:

https://github.com/ShreyasManjunath/rosbag_analysis_task

A README.md file explains the usage of the package when cloned locally. The package also supports docker. The package comes with Dockerfile, docker-compose.yml, startpoint.sh and .env file. The image can be built and used locally, or you can visit the link:

https://hub.docker.com/r/shreyasmanjunath/rosbag_analysis

OR

Use the docker command to get the docker image.

```
$ docker pull shreyasmanjunath/rosbag_analysis
```

If you use the docker image directly, use the docker docker-compose.yml and .env file from the github repository. Edit the .env file fields based on your scenario:

```
ROSBAG_NAME=xyz.bag

BAG_LOCATION=/home/name/folder

OUTPUT_LOCATION=/home/name/folder
```

Use the docker-compose command to spin-up the container:

```
$ docker-compose up -d
```

The report.log will be generated in the output location provided by the user.

Note: If the bag is not found, the report file will be empty with only header info and when docker logs are checked, the error is prompted indicating “Error opening file: <bagfile_name>”.