Node Robotics

ROSBAG ANALYSIS REPORT

April 25, 2022

Nayan Man Singh Pradhan
nayan.pradhan@hotmail.com

Contents

1 Situation

Today is 2nd October. I come to the office and get a call from one of our customers that there was a problem with one of their robots yesterday in a plant. The robot stopped driving between 11:20 and 11:30 in front of a dolly. Luckily, the customer uploads rosbag files to the cloud and I can download them.

2 Approach

First I download the rosbag files to evaluate the error. In the rosbag files, I checked the ros topics and analysed the behaviour of the topics. Since I initially did not know what the topics were publishing, I did a `rostopic echo /topic_names` for multiple topics while playing the rosbag files. I was provided with 10 rosbag files. I went through each rosbag file and analysed the topics. In order for me to understand what was really going on, I visualized the topics using RVIZ. I selected all relevant topics and visualized the robot autonomously driving through the environment. I will discuss visualization in detail in 3.1. I paid special attention to the rosbag files:

- dummy_env-agv-50231.agv-2020-10-01T082312+0200_2020-10-01-11-28-17_37.bag
- dummy_env-agv-50231.agv-2020-10-01T082312+0200_2020-10-01-11-33-17_38.bag.bag

as they were bag files from 1st October between around 11:20 to 11:30 when the robot stopped driving.

Some interesting topics I analyzed were:

- `/cmd_vel`
- `/long_term_slam/error_code`
- `/long_term_slam/state_monitoring/twist_limit`
- `/lts_confidence_marker`
- `/map`
- `/mission_active`
- `/move_base/local_costmap/costmap`
- `/obstacle_collision_filter/obstacle_zone`
- `/odom`
- `/scan`
- `/state_machine_event_log`

- `/stop_signal`
- `/stop_signal_from_left_motor_enable_status`
- `/stop_signal_from_pause_request`
- `/stop_signal_from_right_motor_enable_status`
- `/stop_signal_from_ultrasonic_stop_control`

I then created a ros package containing scripts, launch files, and RIVZ configuration files that would help me better analyze the situation and environment. The ros package can be found in my Github as `node_rosbag_analysis`. The package also contains installation and setup instructions in the `README.md` file.

3 Task 1: Error Analysis

The error occurred in Bagfile dummy_env-agv-50231.agv-2020-10-01T082312+0200_2020-10-01-11-28-17_37.bag. I analyzed the ros topics before, during, and after the error and noticed some specific patterns or messages being published.

3.1 Visualization and Error Evaluation

In order to explain the error to the client, I would launch the `rviz_launch_node` that I created. An example of the RVIZ window running from the `rviz_launch_node` can be seen in Figure 1. This launch file is important because it loads the RVIZ window with the saved preset RVIZ configuration file that already has different relevant messages/markers shown/visualized. The launch file also allows the user/client to easily change the rosbag file and path to rosbag file.

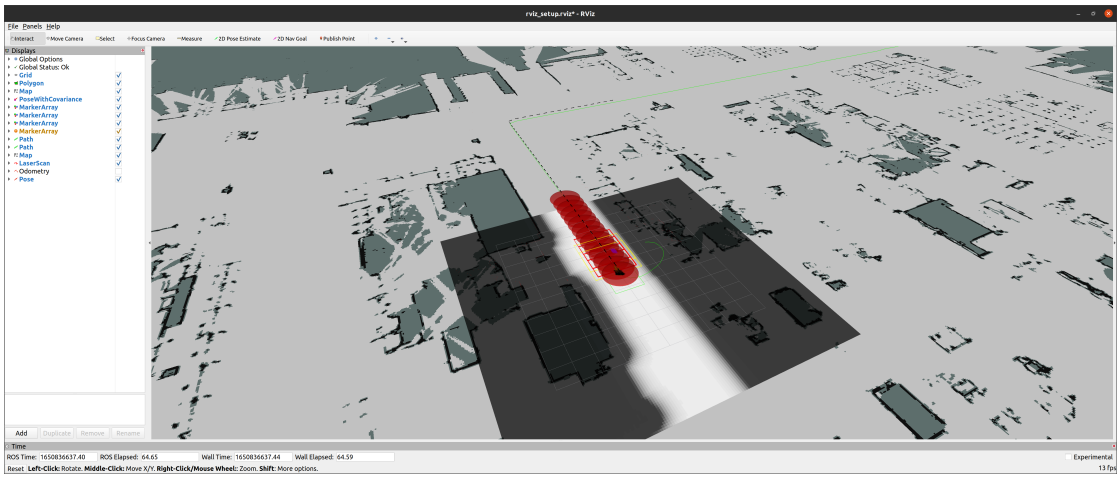
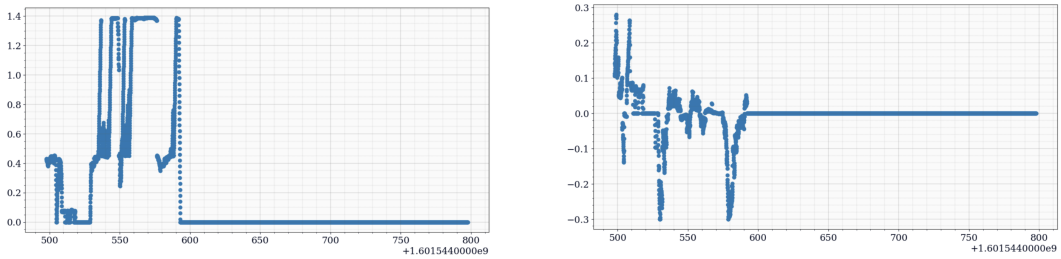


Figure 1: RVIZ Visualization of environment and robot.

I have also uploaded videos of the RVIZ window to Google Drive for easy reference. The uploaded video **video_rviz.webm** is a screen-recording of the robot moving normally and **video_error.webm** is a screen-recording of the robot when the error is seen. It can be seen from the visualization that the robot spins around uncontrollably for a few seconds (around Rosbag Duration: 95.0) and then comes to a stop (around Rosbag Duration: 110.0). Then, the robot does not move for the rest of this bag file and also the consecutive bag file. In order to verify that the robot is not moving, I look into the `/cmd_vel` topic and also use a JupyterNotebook to plot out the linear velocity in x-axis and angular velocity in z-axis. The plots are also shown in Figure 2 for reference. We notice a sudden drop in the `/cmd_vel` message as can be seen from Figure 2. The complete JupyterNotebook codebook can also be found in this package.



(a) Linear velocity in x-axis plot with time.

(b) Angular velocity in z-axis plot with time.

Figure 2: Linear velocity in x-axis and angular velocity in z-axis plot with time.

3.2 Potential Error

I had to look back and forth between the bag file with the error and the normal bag files in order to investigate the potential error. From my investigation, the `/state_machine_event_log` topic was very useful as it published information about what event the state machine is attempting to do, the status of the event, and the outcome as "Init, "Transition", and "Termination" strings. From my inspection, I noticed that when the "Termination" message was "paused" the error had occurred.

The topic `/stop_signal_from_left_motor_status` and topic `/stop_signal_from_right_motor_status` were also extremely interesting. I noticed that the robot had first suddenly started spinning in circles in a counter-clock direction. This led me to believe that the motors on the left-hand side had stopped running for some reason while the motors on the right-hand side were still running. This hypothesis was indeed proven when I observed the `/stop_signal_from_left_motor_status` and `/stop_signal_from_right_motor_status` topics. When the error occurred, the `/stop_signal_from_left_motor_status` was true and `/stop_signal_from_right_motor_status` was false, meaning that the motors on the left had stopped while the motors on the right were still activated causing the robot to spin in circles in a counter-clock wise direction.

While observing the `/stop_signal_from_pause_request` topic, I noticed that the signal was set to true during the error incident. This means that the robot should have completely stopped. Then I noticed that when this topic was set to true, the `/stop_signal_from_left_motor_status` was first set to true, then after a delay the `/stop_signal_from_right_motor_status` was set to true. I believe that this delay and the explanation from the above paragraph caused the robot to spin around for a few seconds and suddenly stop.

I then thought of possible reasons the robot did not re-plan its route and navigate to the target location. In order investigate this, I looked into the topics responsible for SLAM. A simple command `rostopic list | grep slam` and `rostopic list | grep lts` gave me all topics responsible for long term slam. I looked into all the topics and found `/lts_confidence_marker` and `/long_term_slam/state_monitoring/twist_limit` interesting. The `/lts_confidence_marker` topic published the confidence value of the long term slam localization. I noticed that when the robot started suddenly spinning, the value of confidence went down from around 90 to 0. This means that the robot was now not confident of where it was in the environment. This is probably because the laser scanner mounted on the robot scanned a lot of points of the environment while it was spinning and hence could not figure out where it was. The laser scan data can also be visualized in the RVIZ launch node and through the topic `/scan`. I believe that this might be the reason that the robot stopped in the position and the state machine was also paused. I also noticed that the `/long_term_slam/state_monitoring/twist_limit` topic started publishing a twist limit float value of around 0.05 after the robot started spinning and 0.00 when the robot stopped. This might also be what stopped the robot.

From my observations, I believe that the robot might have stopped due to a potential collision with a dolly or because the dolly moved very fast and too close to the robot and the control system of the robot tried to adjust and re-plan the route very quickly. This then might have caused the state machine to pause, which might have given the left and right motors a stop signal. Since there was a delay between the two motors receiving the stop signal, the left motor stopped first while the right kept running. Due to this, the robot moved suddenly in a counter-clock wise manner. Due to this movement, the laser scan picked up a lot of scan data from a very fast moving environment which then decreased the confidence of the long term slam localization algorithm. This finally led to the robot not knowing where it is, not being able to plan out the next path, and hence, stopping completely.

3.3 Potential Solution

According to my observation, a potential solution to this situation could be to:

1. Check whether the `/stop_signal_from_left_motor_status` and `/stop_signal_from_right_motor_status` are receiving messages at the same rate/time.

2. If the robot is stuck/stopped and the long term slam is not confident in the location of the robot, turn the robot around and use the laser scan message `/scan` to get more information about the environment such that it can get a better idea of its position and orientation in the map.

4 Task2: Automatic Error Detection

4.1 Automatic Error Detection and ROS Nodes

After trying to figure out the potential reason for the robot not moving error, I used the topics observed and discussed in the previous section 3.2 to generate ideas for detecting similar errors automatically. The ros package `node_rosbag_analysis` contains two main ros nodes: `real_time_error_detection_node` and `review_error_detection_node` for detecting errors automatically. The `real_time_error_detection_node` subscribes to topics being published in real time and does error detection such that the operator can inspect the robot in real time using RVIZ for visualization. Since this node directly subscribes to the topics, one can comment out the line launching the specific rosbag file and use the script to detect errors in the factory environment. The operator is also allowed to used RVIZ (automatically launched with RVIZ configuration file) such that the operator can inspect the path, laser-scan data, map, and other topics in real time. If an error occurs, a ROS ERROR message is printed in the console. The node `review_error_detection_node` is made such that there is no visualization using RVIZ. The bag file is automatically read based on the topics of interest and messages are handled sequentially. This node also prints a ROS ERROR message if there is an error in the bag file. Since this node does not require RVIZ or other visualization tools, it can be run in any computer using a Docker container with the provided Dockerfile in the ros package.

The `real_time_error_detection_node` subscribes to the ros topics: `/mission_active`, `/state_mission_event_log`, `/cmd_vel`, and `/scan`. The node first checks whether a mission is active or not using the `/mission_active` message. If the mission is not active, the node does not do additional checks. If the mission is active, the node gets the latest state machine event log message using the topic `/state_mission_event_log`. I then use python string manipulation to get the result of the "outcome" from the state machine event log. If the outcome is paused, the node realizes something might be wrong. Then, the node checks whether the linear velocity in x-axis is less than a minimum linear x threshold (which can be updated in the launch file). The default value for `min_linear_x_threshold` is 0.15. Then this node continues to print a ROS ERROR in the console until a linear velocity in x-axis of value greater than the minimum threshold is applied (until the robot moves again).

The `review_error_detection_node` uses the Python Rosbag API to go through the messages that were published by the ros topics: `/mission_active`, `/state_mission_event_log`, and `/cmd_vel`. The node first checks whether the mission is active or not by the

messages from the topic `/mission_active` and prints a rospy loginfo if the mission is inactive. Similar to the previous node, if the mission is active, the node gets the latest state machine event log message from the topic `/state_mission_event_log`. The result of the "outcome" from the state machine event log is determined. If the outcome is paused, the node realizes something might be wrong. Then, the node checks whether the linear velocity in x-axis is less than a minimum linear x threshold (which can be updated in the launch file). The default value for `min_linear_x_threshold` is 0.15. If the linear velocity in x-axis is less than the threshold, the node prints a ROS ERROR in the console.

Information regarding installation and setup of the package can be found in the `README.md` section of the github repository `node_rosbag_analysis`.

4.2 Using Docker Containers

As mentioned before, a Dockerfile is created to help build a Docker container in order to run the package/tool on different machines or even for the customer machine. The full instructions on installation, setup, and launching the node using a Docker container is described in the `README.md` section of the git repository. The docker container is built using `ros:noetic` and not `osrf/ros:noetic-desktop-full`. Hence, nodes that use graphical visualization like `real_time_error_detection_node` using RVIZ is not possible. A separate launch file `docker_review_error_detection_launch.launch` is prepared to launch the ros nodes in the docker container.