Laboratory Report

Group 2: Autonomous Robotics Lab

Robot ID: 08

Day: Wednesday

Group Member: Wilson Zhu

1. Solution Approach

Our group developed three versions of the algorithm to solve the problem of maze exploration. Their logic is similar, which is mainly to keep following the right wall in the whole maze. It behaves like a greedy approach as if ideally if we can keep track with the right wall then eventually we can get back to where it started. However, although the first two versions had good logic, we faced difficulty implementing and the logic wasn't perfect at that time. The first two versions just keep hitting the wall.

Therefore, my role was to rewrite and implement the whole algorithm and refine our idea. Eventually, after hours of hard coding it worked in the simulation. In summary, the updated algorithm is:

- Default state is to go forward, but if there is no space in the front (0 degree) or right (330) the robot will keep turning left until there is enough front and right space.

- Otherwise, if there is enough space and we got a lot of space on the right(300 degree) which indicates there is a turning point at right. Then the robot will start to make a right arc turn. Until there is not more front distance or right (330) distance it will start to turn left.

- Else, the robot will just go straight until it reaches the starting position.

Then, I transferred this algorithm to code and I soon found out that the escape_range variable and get direction state are not really useful for the wall_follwer so I just abandoned them.

Moreover, although with the new algorithm, our robot can safely make turns and explore in the gazebo_home simulation without bumping against the wall most of the time. We still need to tune and stop the robot when it drives back to the starting point. Daniel makes crucial adjustments to the tuning in angle of checking right and right distance which significantly improve the efficiency of the robot in turn point and ensure the robot to leave the left state in a relatively straight position. Lachlan discovered how to utilise the odometry call back each time to record the start position and monitor the current position of the robot which is also critical in solving the stop at the original point task.

2. ROS2 Packages Utilised and Rationale

The following ROS2 packages were chosen:

Gazebo: This is the simulation tool we used to test the correctness of the algorithm and tuning when there were groups testing in the maze or there was connection failure to the robot. Although it is not very accurate, it still contributes to our project.

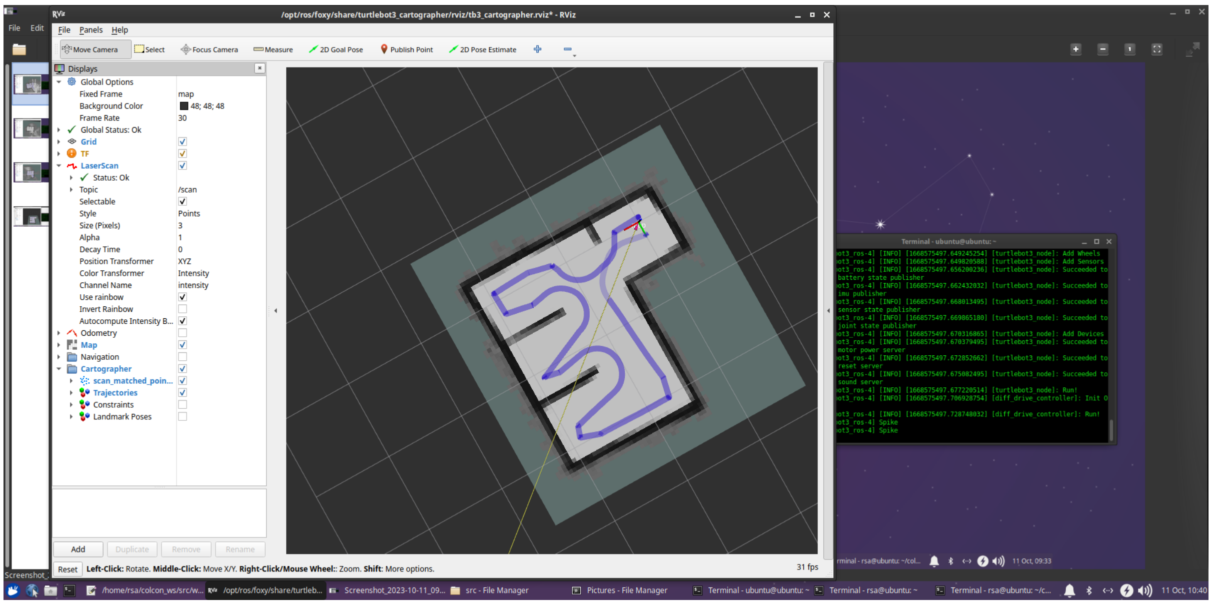
SLAM/Cartographer: For simultaneous localization and mapping, this package was chosen as it can record the map of the maze, trajectory of the movement and the location the robot is in. It helps to achieve multiple tasks in the demo.

Bring_up: This package was used to connect the robot which is the most essential package in our project.

Odometry: We use this package to keep getting the robot position so we can terminate it if it reaches the start point.

3.Performance Evaluation

Here is the map and the video of our final run.



```
Global Options
  Fixed Frame                  map
  Background Color             48; 48; 48
  Frame Rate                   30
  Global Status: Ok
Grid                           ✓
TF                             ✓
LaserScan                      ✓
  Status: Ok
  Topic                        /scan
  Selectable                   ✓
  Style                        Points
  Size (Pixels)                3
  Alpha                        1
  Decay Time                   0
  Position Transformer         XYZ
  Color Transformer            Intensity
  Channel Name                 intensity
  Use rainbow                  ✓
  Invert Rainbow
  Autocompute Intensity B...   ✓
Odometry
Map                            ✓
Navigation
Cartographer                   ✓
  scan_matched_poin...         ✓
  Trajectories                 ✓
  Constraints
  Landmark Poses
```

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.

```
ot3_ros-4] [INFO] [1668575497.649245254] [turtlebot3_node]: Add Wheels
ot3_ros-4] [INFO] [1668575497.649820588] [turtlebot3_node]: Add Sensors
ot3_ros-4] [INFO] [1668575497.656200236] [turtlebot3_node]: Succeeded to
 battery state publisher
ot3_ros-4] [INFO] [1668575497.662432032] [turtlebot3_node]: Succeeded to
 imu publisher
ot3_ros-4] [INFO] [1668575497.668013495] [turtlebot3_node]: Succeeded to
 sensor state publisher
ot3_ros-4] [INFO] [1668575497.669865180] [turtlebot3_node]: Succeeded to
 joint state publisher
ot3_ros-4] [INFO] [1668575497.670316865] [turtlebot3_node]: Add Devices
ot3_ros-4] [INFO] [1668575497.670379495] [turtlebot3_node]: Succeeded to
 motor power server
ot3_ros-4] [INFO] [1668575497.672852662] [turtlebot3_node]: Succeeded to
 reset server
ot3_ros-4] [INFO] [1668575497.675082495] [turtlebot3_node]: Succeeded to
 sound server
ot3_ros-4] [INFO] [1668575497.677220514] [turtlebot3_node]: Run!
ot3_ros-4] [INFO] [1668575497.706928754] [diff_drive_controller]: Init O

ot3_ros-4] [INFO] [1668575497.728748032] [diff_drive_controller]: Run!
ot3_ros-4] Spike
ot3_ros-4] Spike
```

https://youtu.be/QFcl_gaA8SA

The performance of our autonomous software was, on the whole, satisfactory. During the live demonstration:

- Robots efficiently explore the whole maze and scan the whole maze accurately.
- It didn't bump into the wall because we left enough front space and right space for the robot to turn.
- It seeks every turning point as the right camera detects large right spaces effectively.
- The greedy algorithm of keeping following the right wall successfully returns to the start point and our odometry checks work and pauses the robot.
- However, the left turn adjustment didn't work as we expected. It made the robot turn a bit too much left and made the trajectory not smooth enough. It is probably because the right_center(330) camera and distance tuning is not good enough in the real condition.

4. Software Limitations

Algorithm defection:

There is still a defect in the algorithm I wrote and we found that after the final demo. There are multiple robots that are so close to the left and the robot should escape the right arc turning state earlier. Otherwise, the robot may hit the left wall in a harsher and narrower turning point which is really likely to happen in a USAR environment.

Difficulties in utilise hardware in harsh environment:

Moreover, the robot relies highly on LIDAR data to calculate distance which may be not reliable if there are mirror and froggy environments which is also possible in a USAR task. Likewise, the robot also faced difficulties in complex and thin object, In simulation our robot is very easily crash into thin mailbox and polygonal object which again reflects the limitations of our robots.

Dynamic Obstacles Handling:

Our robot heavily relies on checking the front and right distance every 10ms. But if there are dynamic obstacles, our robot will not only detect and avoid it until it gets really close, but also be unacceptable slow in escaping the dynamic obstacles. Without solving this issue our robot cannot facing the complex, dynamic changing and unexpected USAR environment.