SLAM AND PATH PLANNING

MY INTERNSHIP EXPERIENCE

By: Hossam Eldin Omar
Program: Robotics internship
Provider: Ejada
Supervised By: Dr. Izzeddin Teeti,
Eng. Ahmed Sameh

ABOUT EJADA

Ejada Systems Ltd. is a leading Saudi IT company focused on digital innovation. Its newly formed robotics team explores advanced technologies like robotics vision and control, Reinforcement Learning, SLAM, and path planning. The team supports hands-on learning through internships and real-world robotics projects, reflecting Ejada’s commitment to future-focused tech.

WHAT IS SLAM?

SLAM stands for Simultaneous Localization and Mapping (sometimes called Synchronized Localization and Mapping). It is the process of mapping an area while keeping track of the location of the device within that area. This is what makes mobile mapping possible, allowing the digitization of large areas in much shorter spaces of time. SLAM systems simplify data collection, providing an avenue for scanning outdoor or indoor environments.

THE SLAM PROBLEM

SLAM is difficult — it suffers from the “chicken or egg problem.” In order to accurately localize, we need good maps, but to have good maps, we need to accurately localize. There is a lot of extensive research for SLAM, combining techniques from engineering, sensor fusion, statistics, probability, computation, computer science, and more. And there are many SLAM algorithms and variants

TYPES OF ALGORITHMS AND APPROACHES TO **SLAM**

1

Graph SLAM

Represents the robot's poses and landmarks as nodes in a graph and optimizes the whole trajectory using constraints.

2

EKF SLAM

Uses the Extended Kalman Filter to estimate the robot's position and landmark locations, assuming Gaussian noise.

3

Fast SLAM

Combines particle filters with EKF to efficiently handle large environments by factoring robot poses and landmarks.

4

Visual SLAM (ORB-SLAM)

Uses cameras (RGB or RGB-D) to perform SLAM by extracting visual features from the environment.

5

2D Lidar SLAM (Graph-based SLAM)

Uses 2D laser scans to build flat, floor-plan-like maps—common in indoor navigation.

6

3D Lidar SLAM (Graph-based SLAM)

Uses 3D laser scanners to generate detailed 3D maps—ideal for outdoor or complex environments.

CONSTANT IMPROVEMENT OF SLAM

Testing SLAM technology in notoriously difficult scanning environments, such as repetitive corridors, smooth-sided tunnels, and spaces with moving people, helps to refine, hone, and push its limit. The result is a constantly improving SLAM algorithm, one that is so robust that it now works equally well in outdoor open environments as it does indoors.

WHAT ARE THE BENEFITS OF USING SLAM?

SLAM-based systems are inherently mobile - they are at their best when used on the move. With a SLAM mobile mapping system, it's possible to simply walk through an environment building a digital map as you go, saving time and providing a cost-effective solution. A SLAM-based mobile mapping system is 10 times faster at acquiring data.

What is Path Planning?

Path planning is the process by which a robot determines a safe, collision-free, and efficient route from its current location to a target destination. It allows the robot to navigate through complex environments by avoiding obstacles and following a calculated path based on a global map or real-time sensor data.

Commonly Used Path Planning Algorithms in SLAM

Global Path Planning Algorithms

Dijkstra's Algorithm

A graph-based algorithm that finds the shortest path by exploring all possible paths.

A* Algorithm

A heuristic-based search algorithm that finds the shortest path from a start point to a goal point.

RRT Algorithm

A sampling-based algorithm that explores the space by randomly sampling points and connecting them to the nearest node.

Local Path Planning Algorithms

TEB Algorithm

An optimization-based local planner that generates smooth, time-efficient, and feasible trajectories in real time.

DWA Algorithm

A local path planning algorithm that considers the robot's dynamics and constraints to compute a feasible trajectory.

RPP Algorithm

The Regulated Pure Pursuit algorithm makes incremental improvements on state of the art by adjusting linear velocities with particular focus on safety in constrained and partially observable spaces

FROM THEORY TO PRACTICE

- Now that we’ve seen the theory behind SLAM and path planning, I’ll walk you through how I applied these concepts during my internship with Ejada’s robotics team.

- The following slides highlight the tools, workflows, and challenges I encountered while working with real robotic systems in simulation.

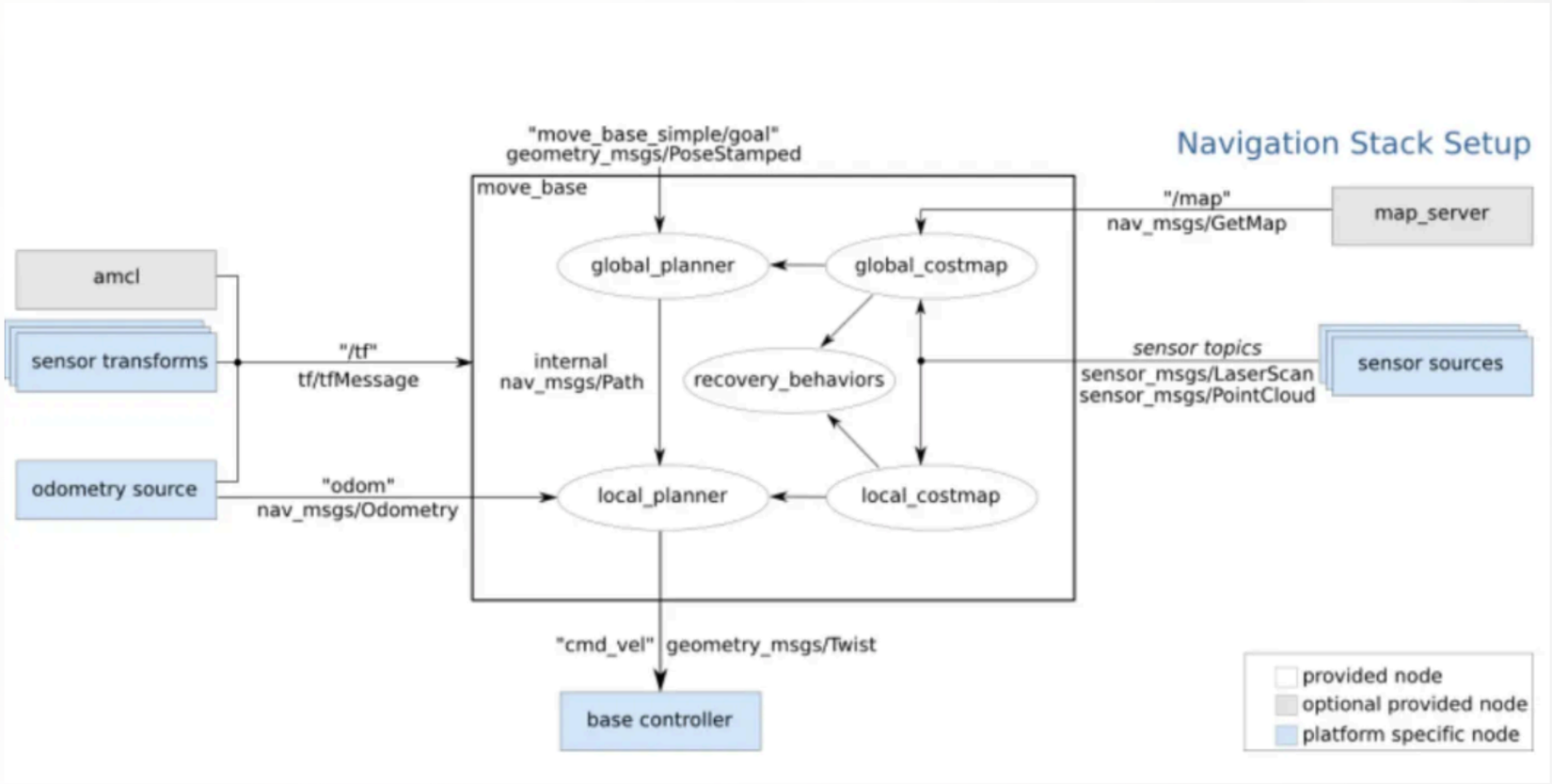
TOOLS AND FRAMEWORKS

- **ROS2 Humble** – middleware for robot software integration
- **Gazebo** – 3D simulator for testing robots in virtual environments
- **RViz** – for visualization of sensor data and robot state
- **RTAB-Map & SLAM toolbox** – for visual SLAM and map generation
- **CHAMP** – for quadruped robot control and locomotion
- **Nav2** – for path planning and navigation stack in ROS2
- **TurtleBot3** – mobile robot platform used for testing SLAM and navigation

SLAM & NAVIGATION WORKFLOW

This diagram represents a typical SLAM and navigation stack setup in ROS.
It illustrates how components like localization, mapping, path planning, and control work together to enable autonomous navigation.

In the following slides, I'll walk through the specific steps I took — from generating a map using SLAM, to configuring the navigation stack, and finally enabling the robot to navigate autonomously in the environment.



SLAM & NAVIGATION WORKFLOW

STEP-BY-STEP

1

Launch the Simulation Environment

I launched the robot (TurtleBot3 or CHAMP) in a Gazebo world designed for indoor navigation. This gave me a virtual environment to test SLAM and path planning.

2

Run SLAM Node (RTAB-Map or SLAM Toolbox)

I started the RTAB-Map SLAM node to begin generating a map while the robot explored the environment. The SLAM system used depth or RGB-D data to localize and map in real-time.

3

Teleoperate the Robot to Explore

Using a keyboard teleop, I manually navigated the robot through the environment to help SLAM build a complete map.

SLAM & NAVIGATION WORKFLOW
STEP-BY-STEP

4

Save the Map

Once mapping was complete, I saved the map using the map_server tool, which generated .pgm and .yaml files for the static map.

5

Launch the Navigation Stack

After mapping, I launched the navigation stack (nav2) and loaded the saved map using the map_server.

6

Set Initial Pose and Goal

I manually set the robot's initial pose in RViz using the AMCL localization system, then gave it a goal using 2D Nav Goal.

PLANNER BENCHMARK COMPARISONS (GLOBAL PLANNER COMPARISON – A* VS DIJKSTRA)

Metric	A*	Dijkstra
Path Length	4.3 m	4.5
CPU Load	Lower	Slightly Higher
Path Shape	Direct	Conservative

Notes:

- A* was generally faster and produced more direct paths because it uses heuristics (e.g., Euclidean distance or Manhattan distance).
- Dijkstra explores all paths uniformly, making it slower and more conservative.

PLANNER BENCHMARK COMPARISONS (LOCAL PLANNER COMPARISON – DWB VS RPP)

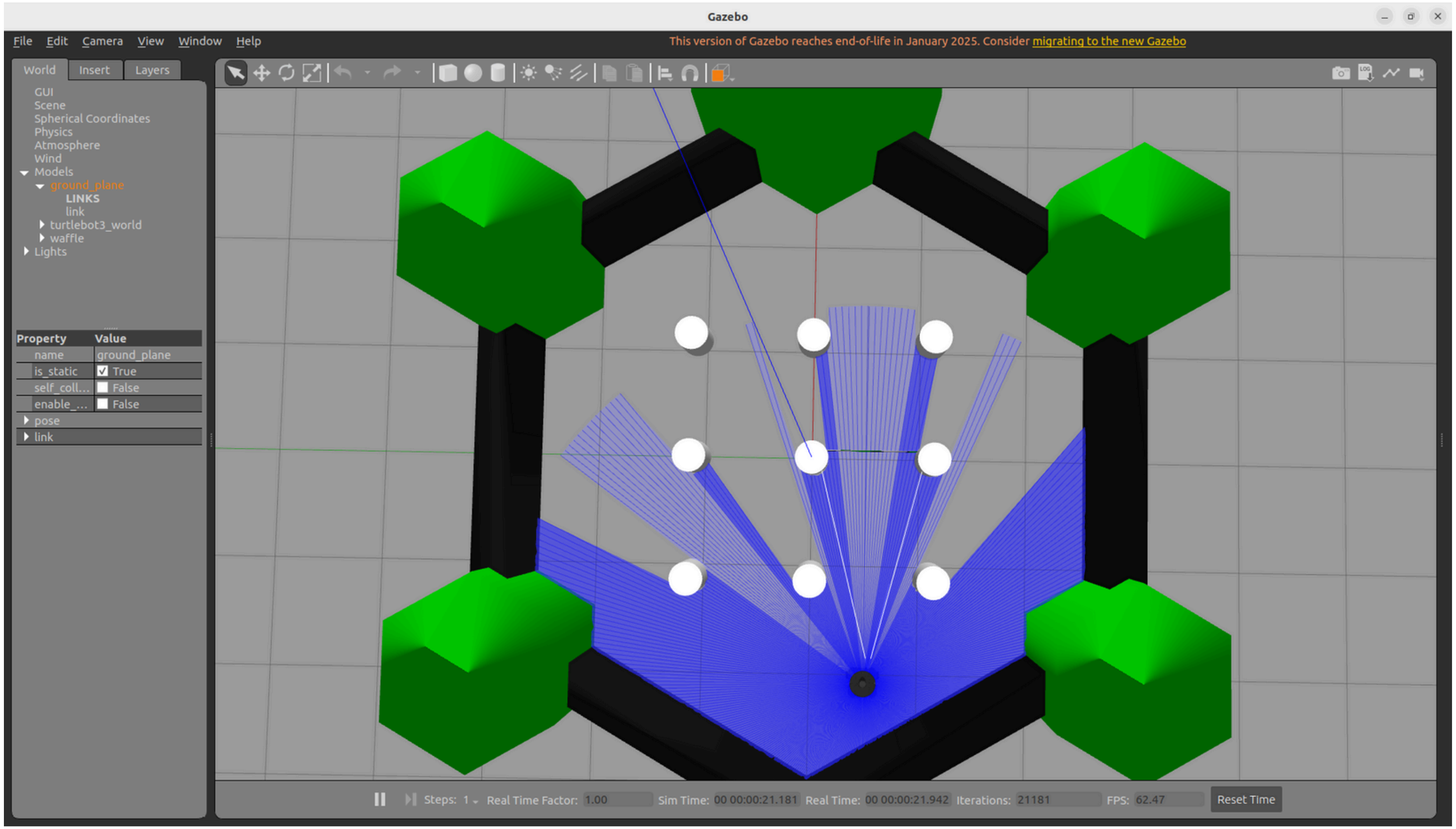
Metric	DWA	RPP (Regulated Pure Pursuit)
Obstacle Avoidance	Very reactive (uses critics)	Smooth, less responsive
Path Smoothness	Moderate	Very smooth
CPU Load	Higher	Lower
Tuning Complexity	High (many parameters)	Low (simpler to tune)

Notes:

- DWA excels in tight spaces but is complex to configure.
- RPP is smoother and simpler but less effective with dynamic obstacles.

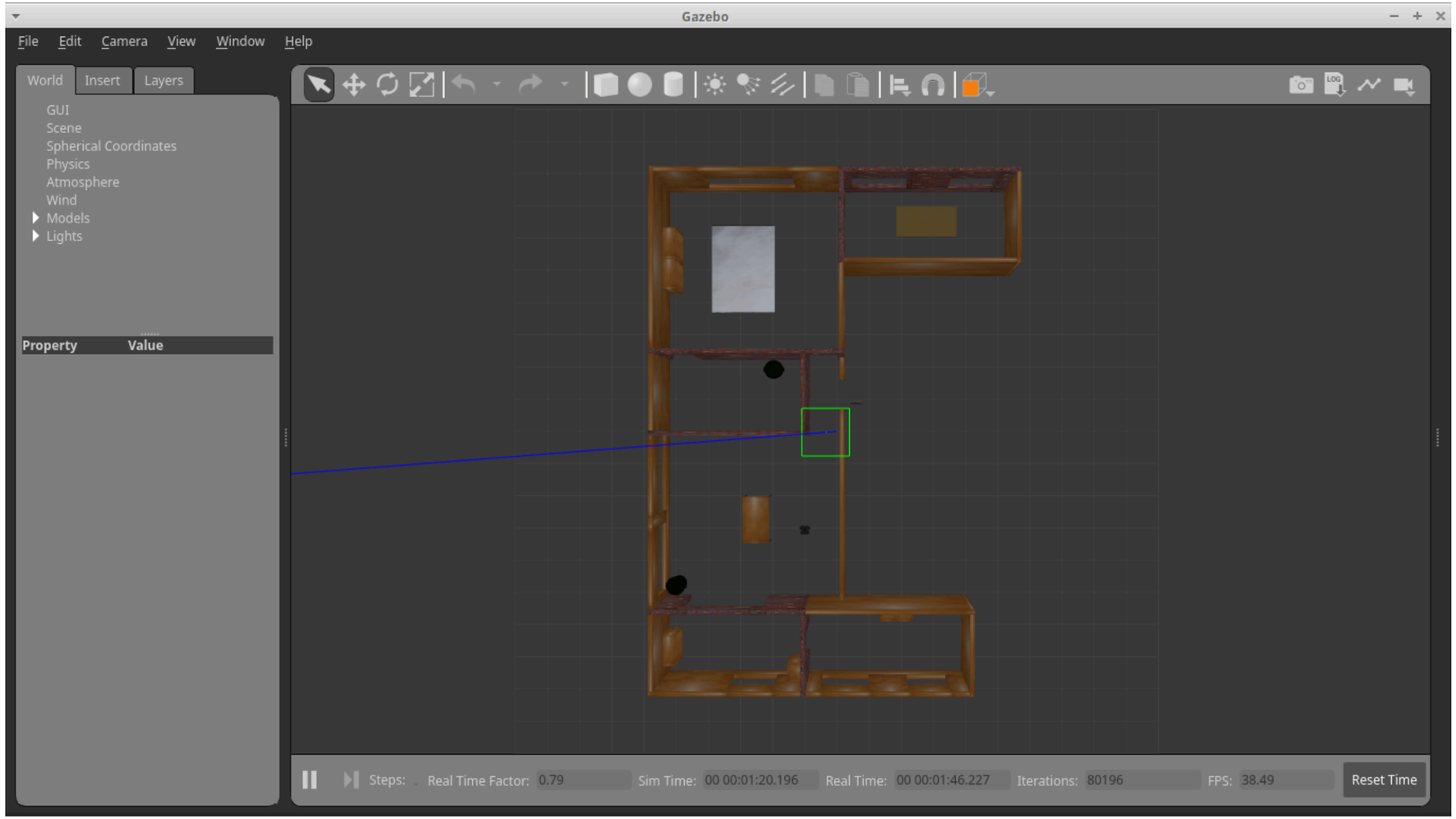
SIMULATION ENVIRONMENTS

Turtlebot3 World



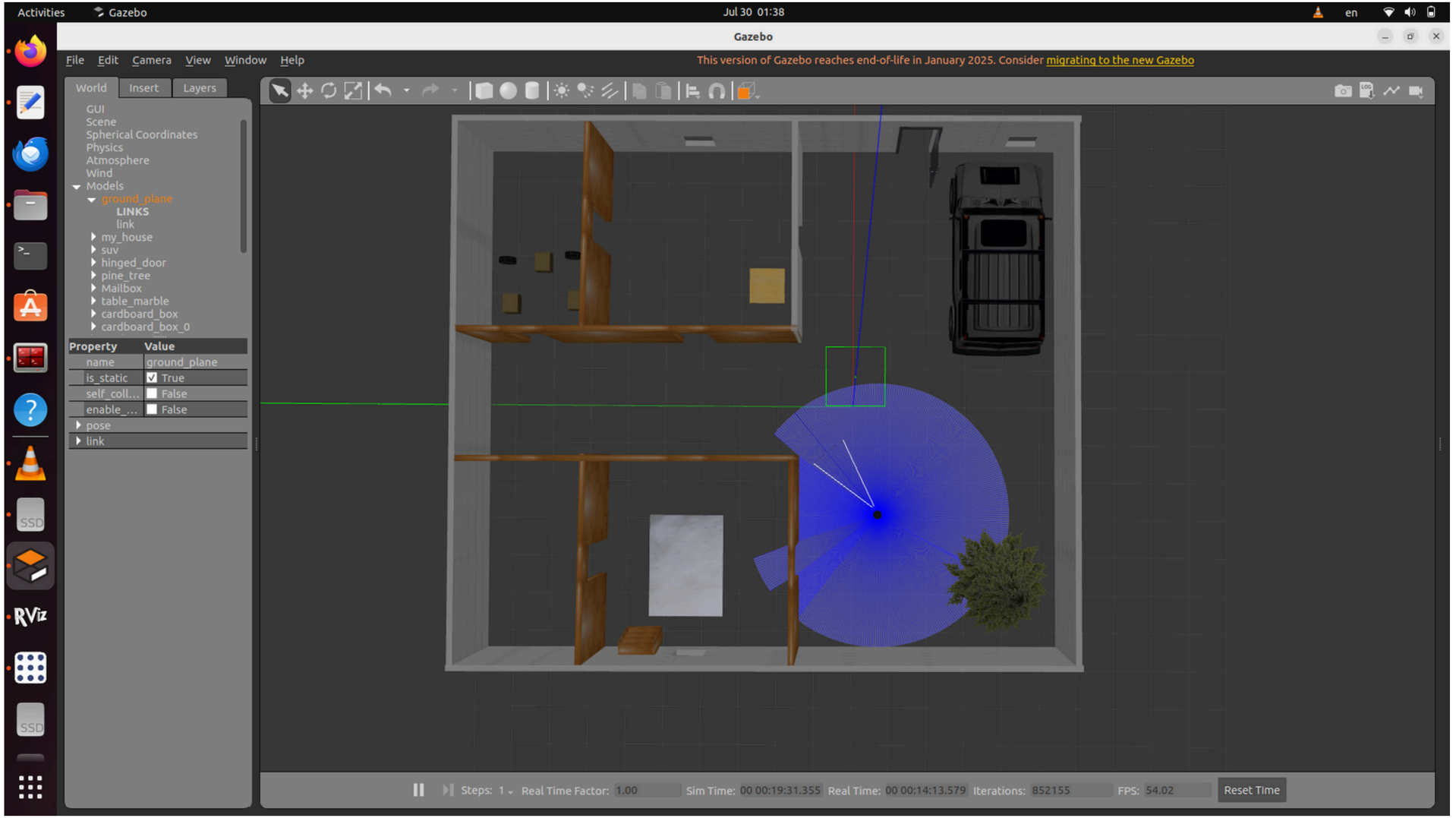
SIMULATION ENVIRONMENTS

Turtlebot3 House



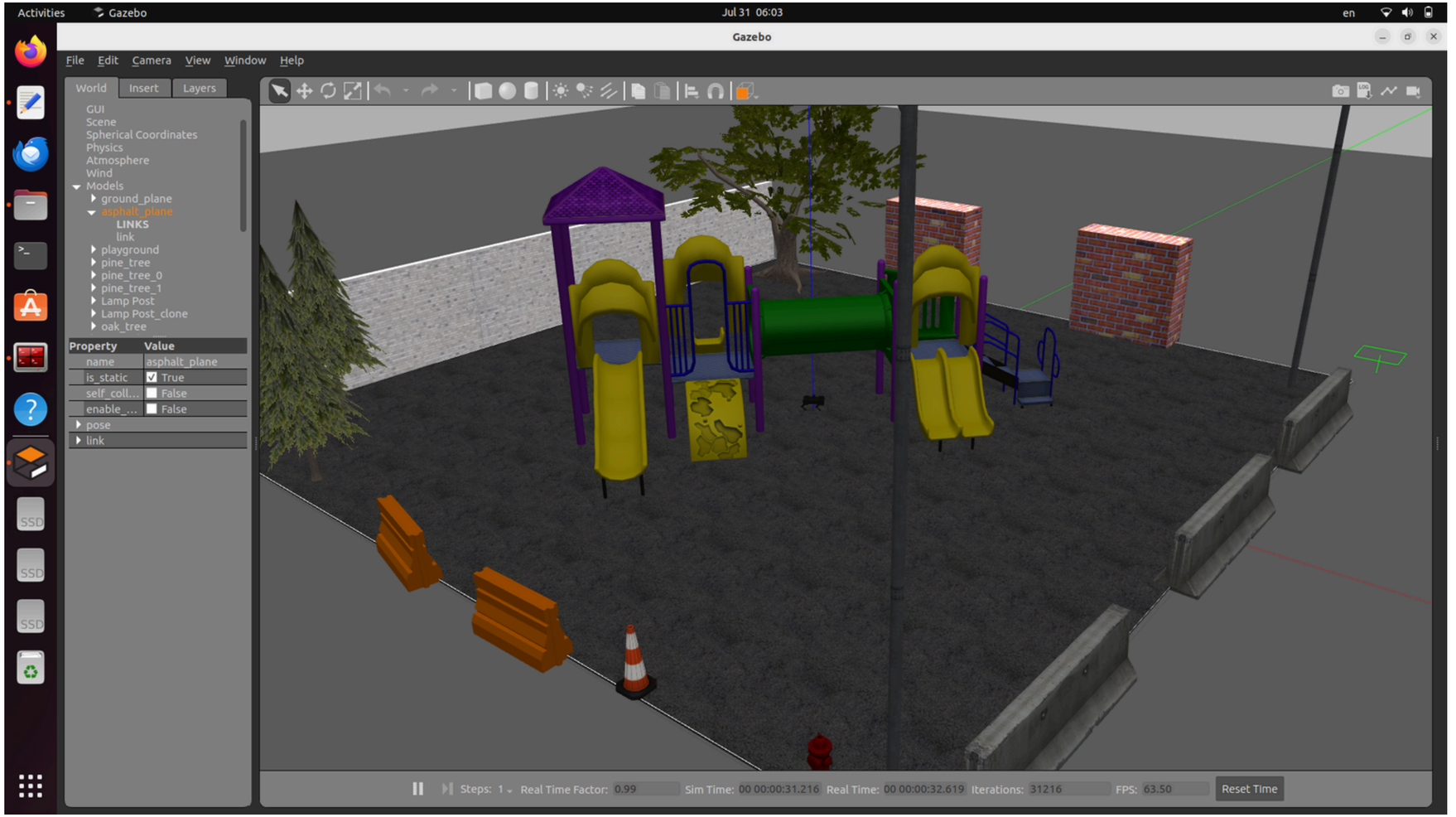
SIMULATION ENVIRONMENTS

Custom world with turtlebot3 robot



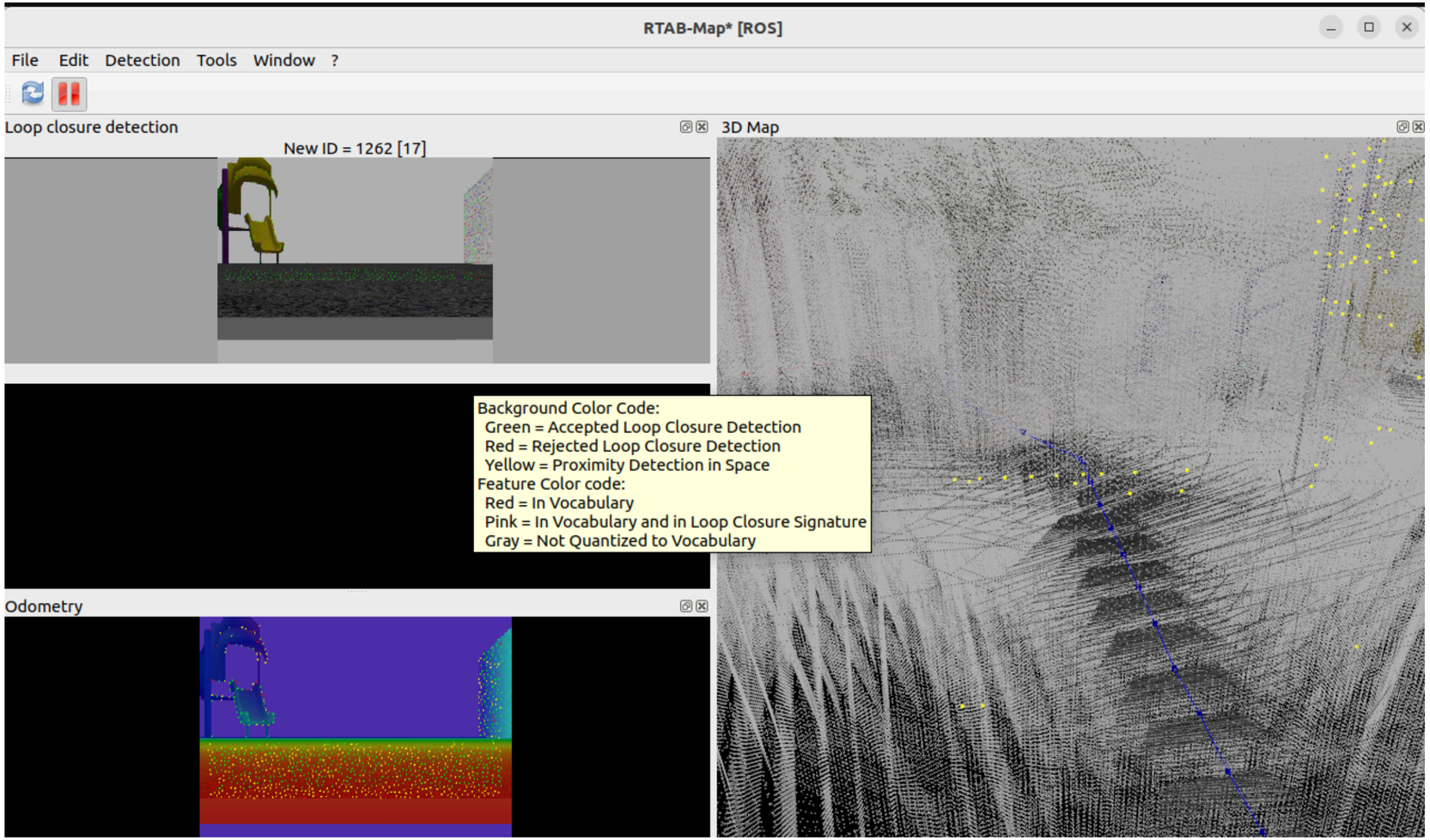
SIMULATION ENVIRONMENTS

Playground world with champ robot



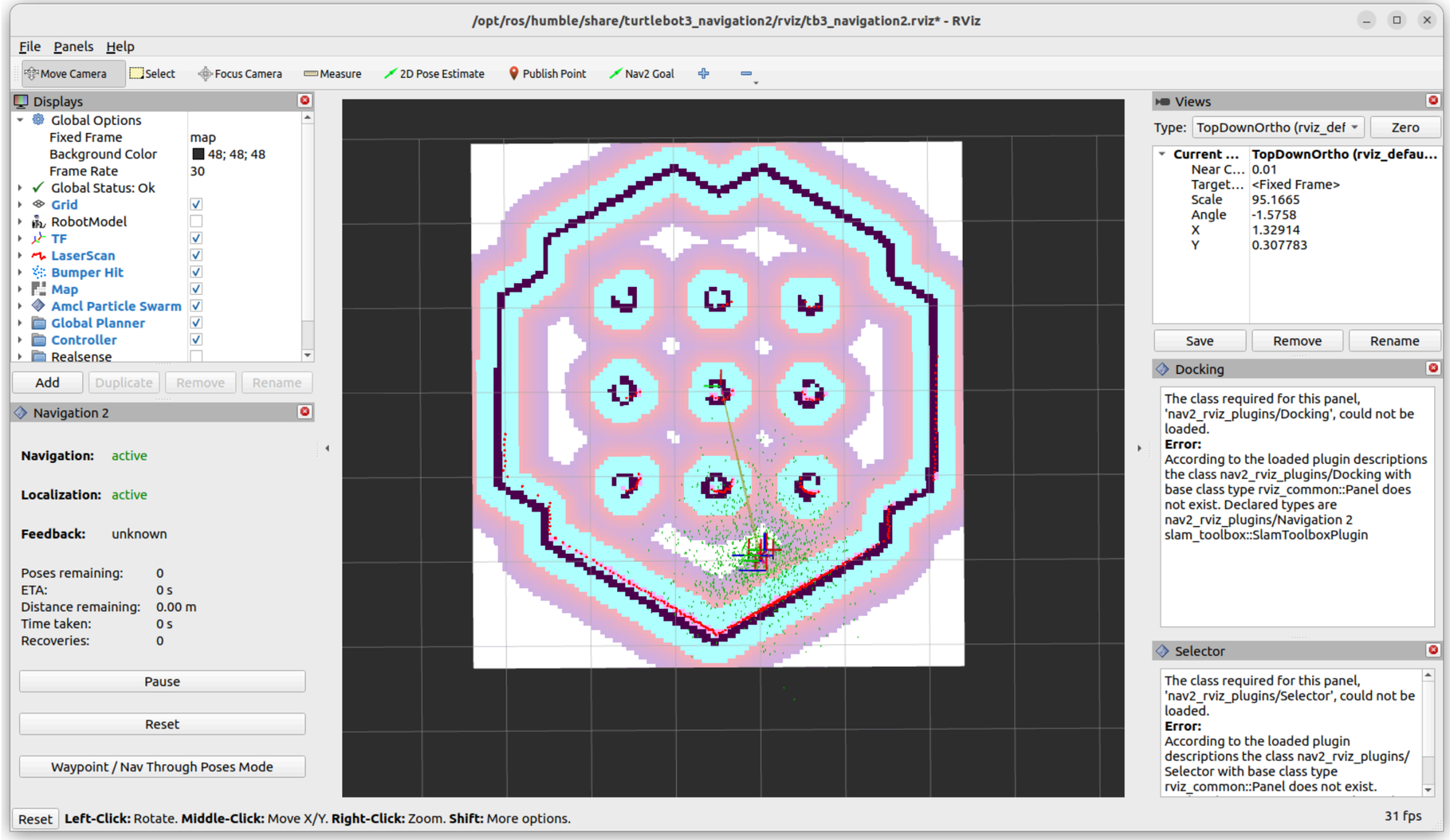
MAPS GENERATED DURING SLAM

Vslam with 3D map generation



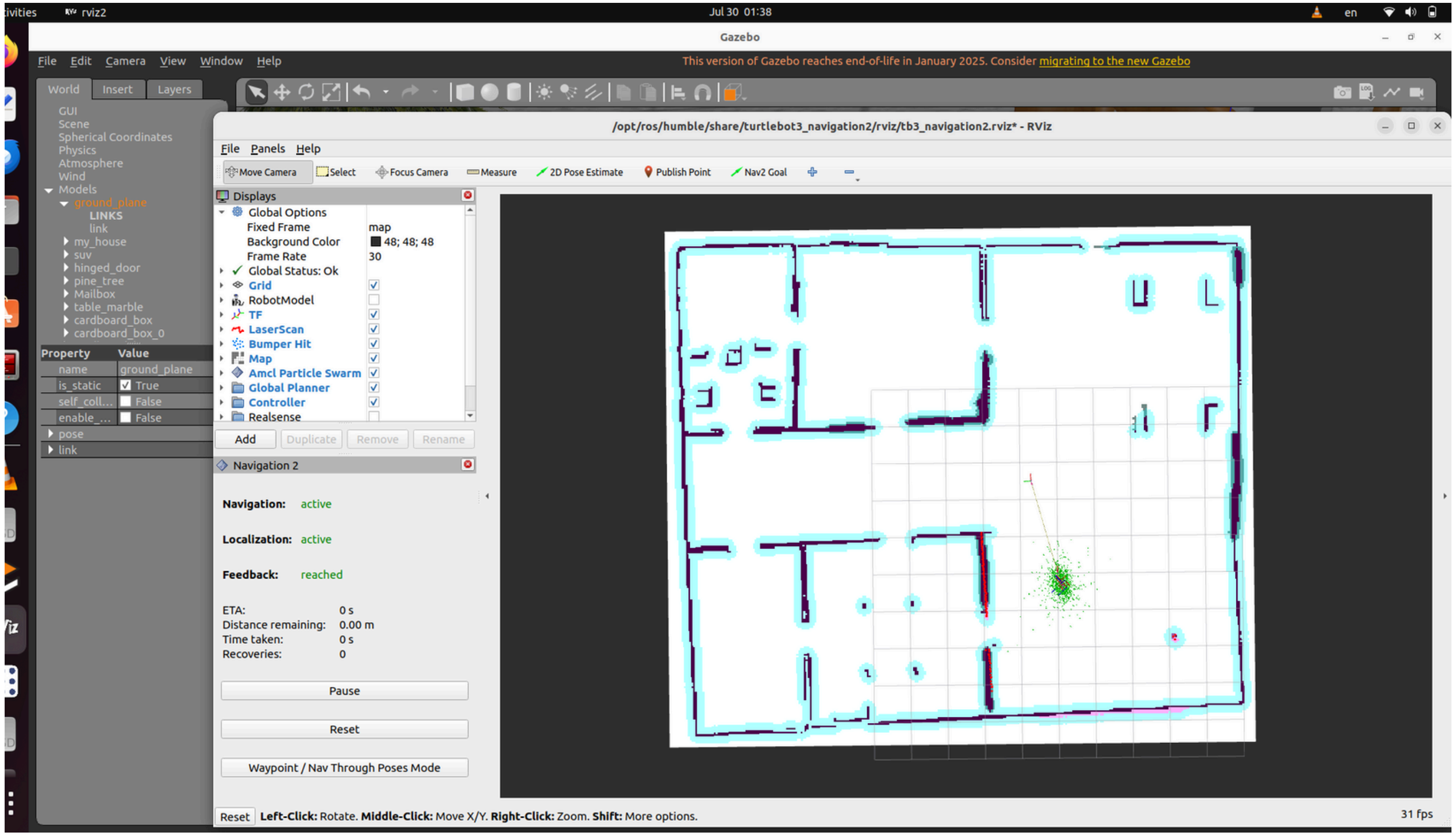
MAPS GENERATED DURING SLAM

2D-LIDAR Map

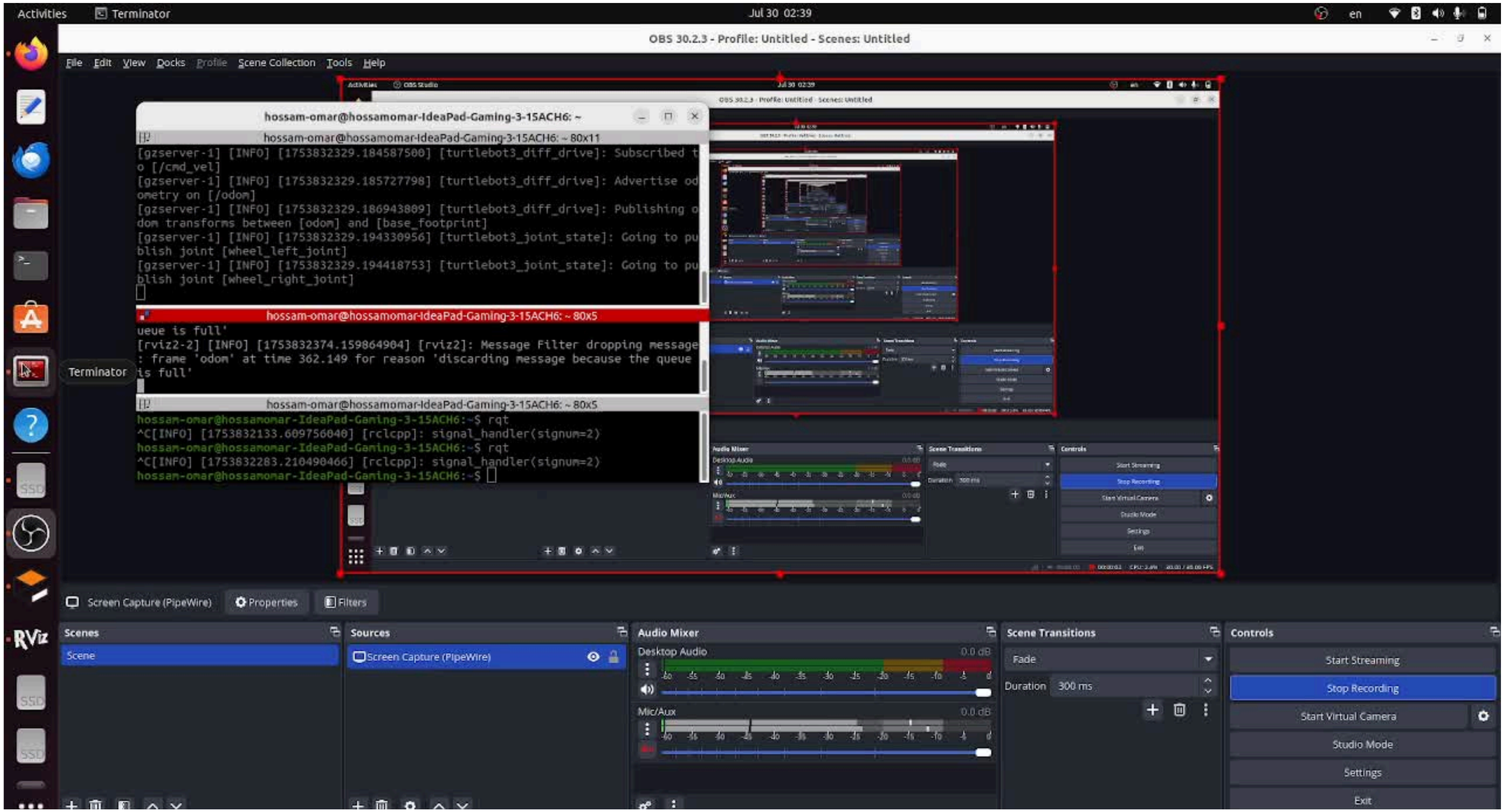


MAPS GENERATED DURING SLAM

2D-LIDAR Map



NAVIGATION IN ACTION



RESOURCES & REFERENCES

- **ROS 2 Documentation (Humble) –** https://docs.ros.org/en/humble
- **Nav2 Navigation Stack –** https://navigation.ros.org
- **RTAB-Map ROS –** https://github.com/introlab/rtabmap_ros
- **FARO Article: What is SLAM –** https://www.faro.com/en/Resource-Library/Article/What-is-SLAM
- **Medium Article: SLAM & Navigation –** https://medium.com/@mykulrizzo/ros-navigation-slam-basics-1a6c52a31b3
- **Medium Article: Types of SLAM Algorithms –** https://medium.com/@nahmed3536/the-types-of-slam-algorithms-356196937e3d
- **Dynamic Window Approach Paper – Fox et al. (1997)** https://www.ri.cmu.edu/pub_files/pub1/fox_dieter_1997_1/fox_dieter_1997_1.pdf

Conclusion

Throughout this internship, I explored the full SLAM and navigation pipeline—from map generation to autonomous path planning. I worked with tools like SLAM Toolbox, RTAB-Map, and Nav2, and gained hands-on experience with simulation environments and benchmarking different planning algorithms.

This journey not only deepened my understanding of robotics but also sharpened my research, debugging, and system integration skills—skills I'll carry forward into future robotics projects.

THANK YOU

PRESENTED BY HOSSAM ELDIN OMAR