CSCI 395

Professor Raj Korpan

Lab 5

Patrick Lin, Lameya Mostafa, Lina Tran

Methods: Briefly describe the process your group followed for each of the three parts:

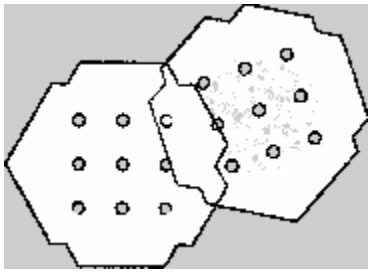
Part 1: How you generated the maps, including your strategy for the high-quality vs. low-quality runs.

To generate high-quality maps, the ideal strategy was to navigate each area of the room at a consistent, moderate speed. During turns, the robot was also kept at a slow and steady pace to allow SLAM sufficient time to perform accurate state estimation. In practice, some errors occurred - particularly turning too quickly and skipping certain areas, such as beneath the table, which were mistakenly assumed to be blocked obstacles.

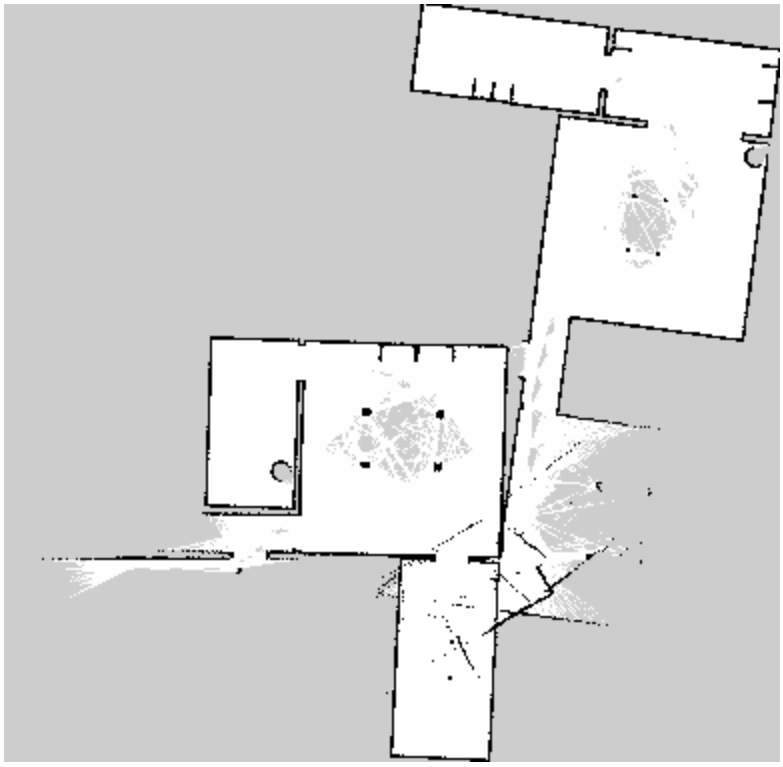
For the low-quality map, the strategy intentionally involved rapid movement through the environment with minimal coverage of each room. The robot moved at inconsistent speeds and executed abrupt turns, preventing SLAM from maintaining accurate localization. As a result, the resulting map contained incomplete rooms and noticeable misalignments.

Results & Visuals: Present your findings clearly. This section must include:

Screenshots of all four final maps from Part 1.
World Map

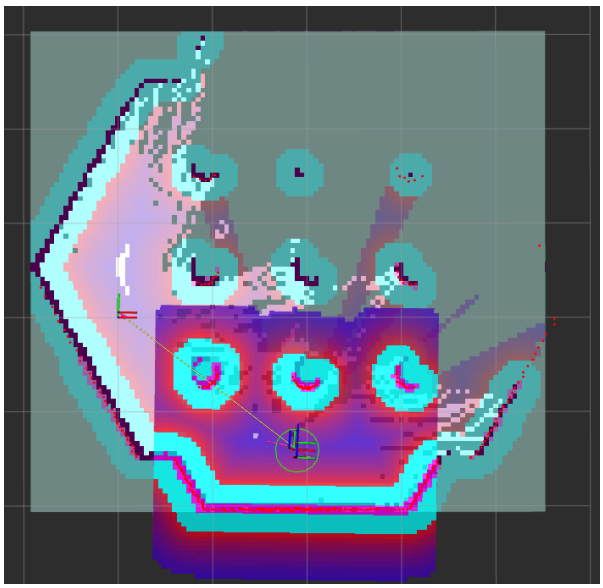
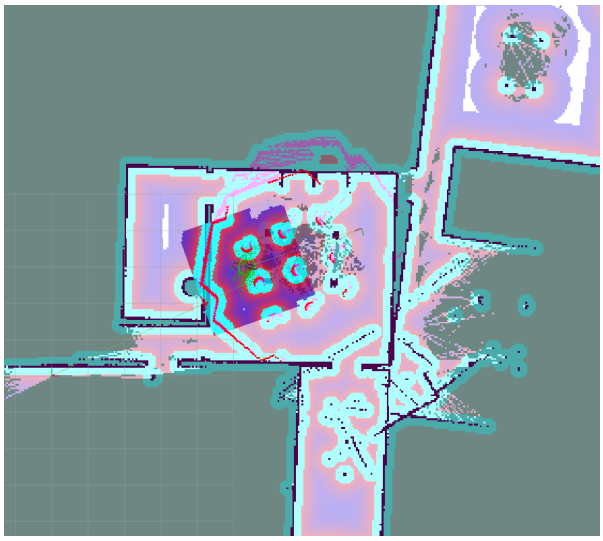


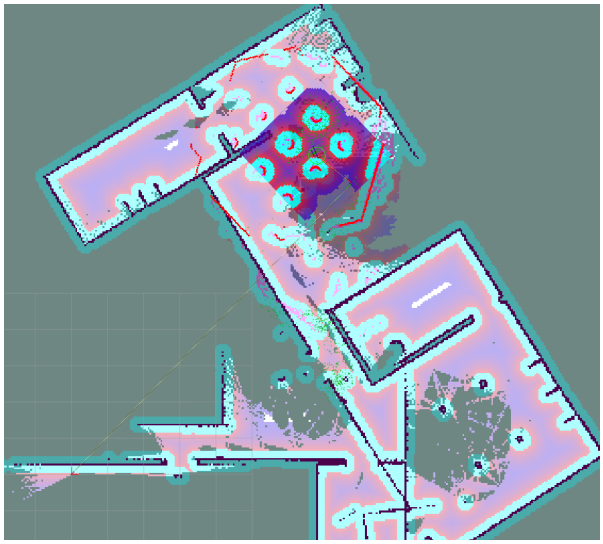
House - Good Version



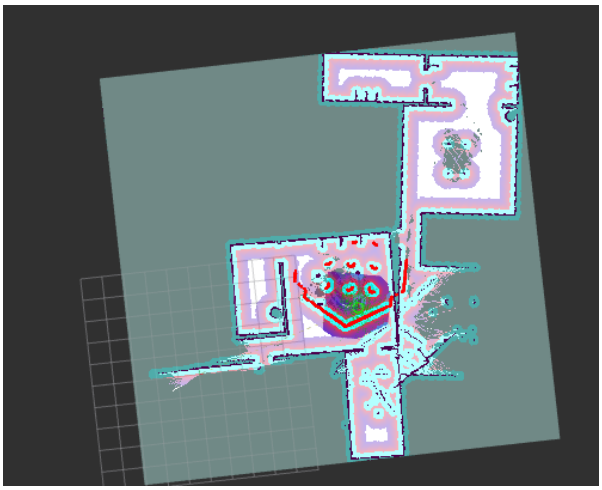
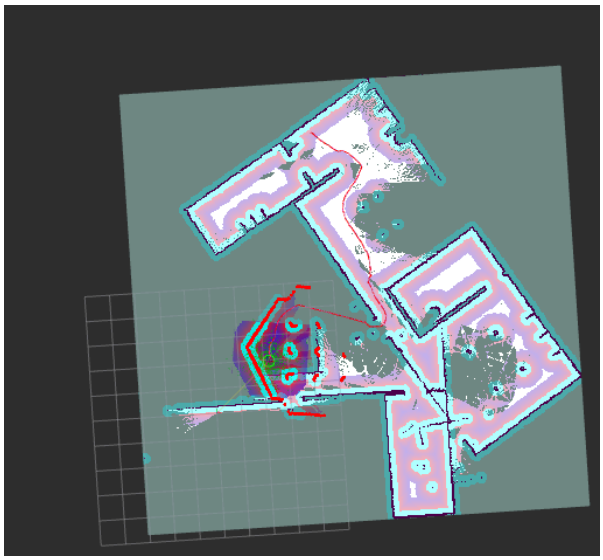
House - Bad

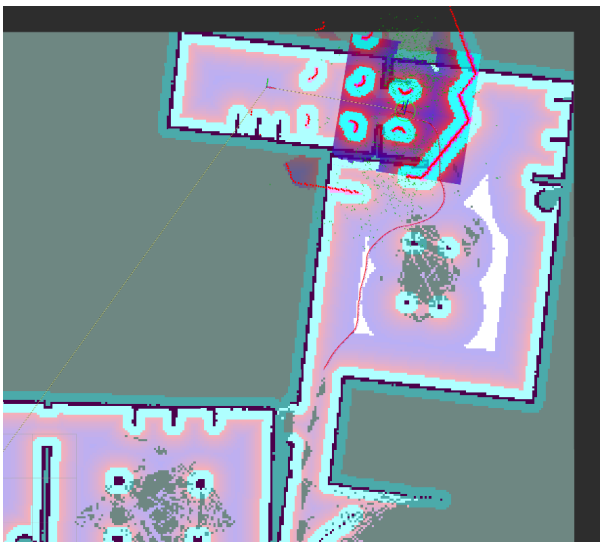
Part 3: Navigate in Each World

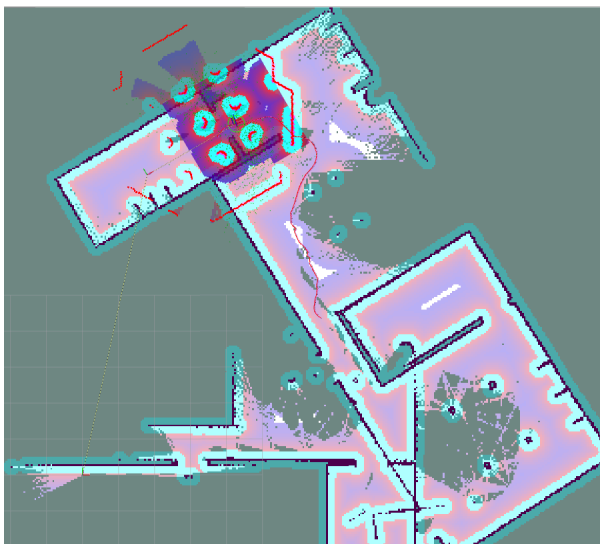
world_map (completed goal)	
good_house (completed goal)	

bad_house (completed goal)	

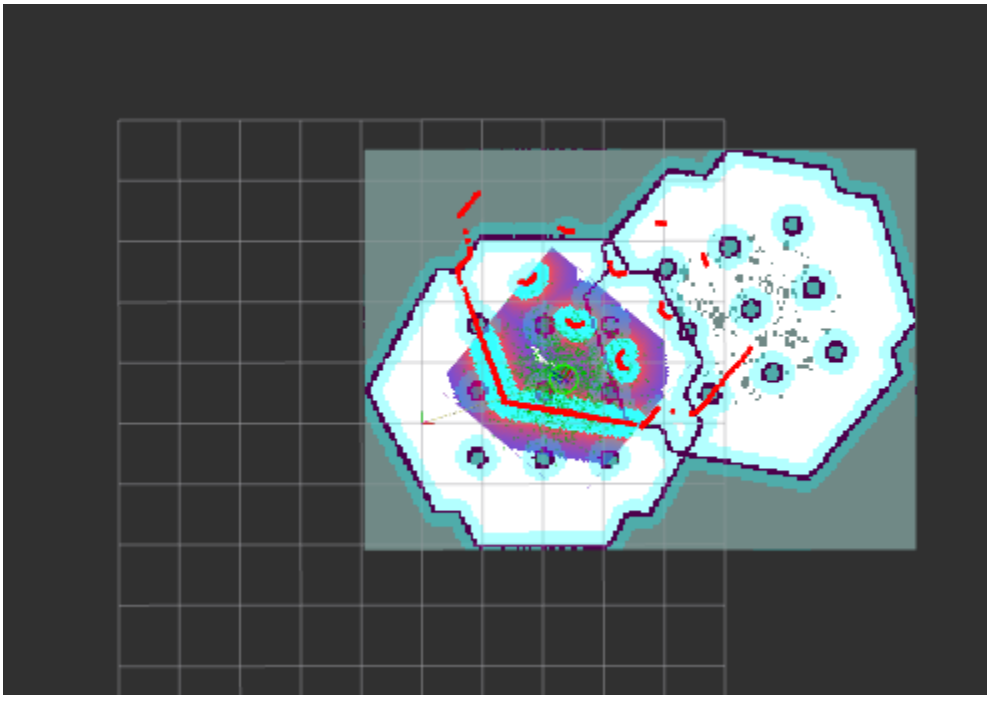
Side-by-side screenshots of the navigation attempts in the good vs. bad house maps from Part 2.

good_house	bad_house
	

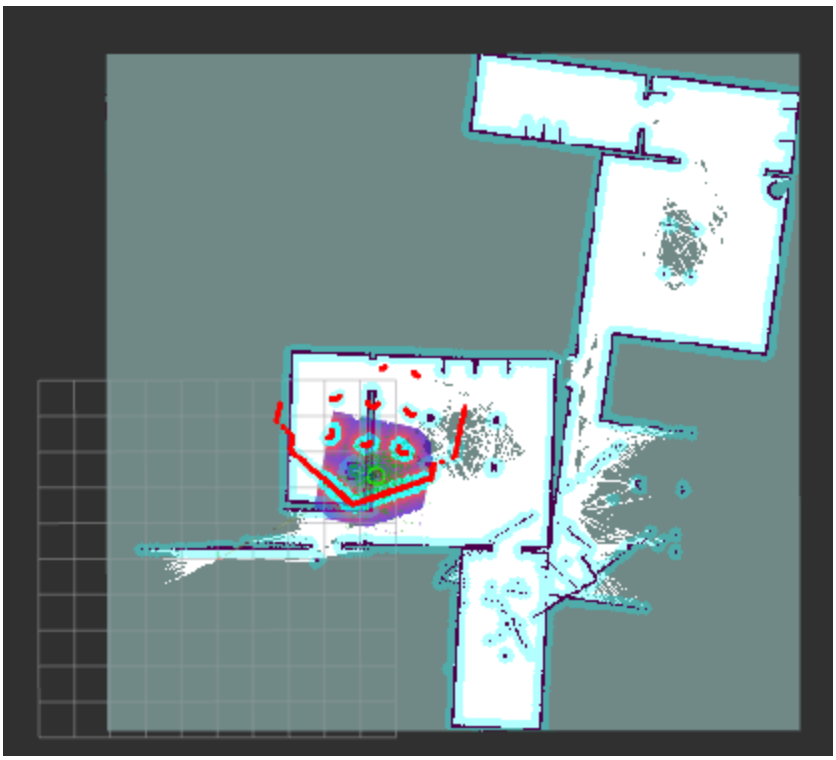




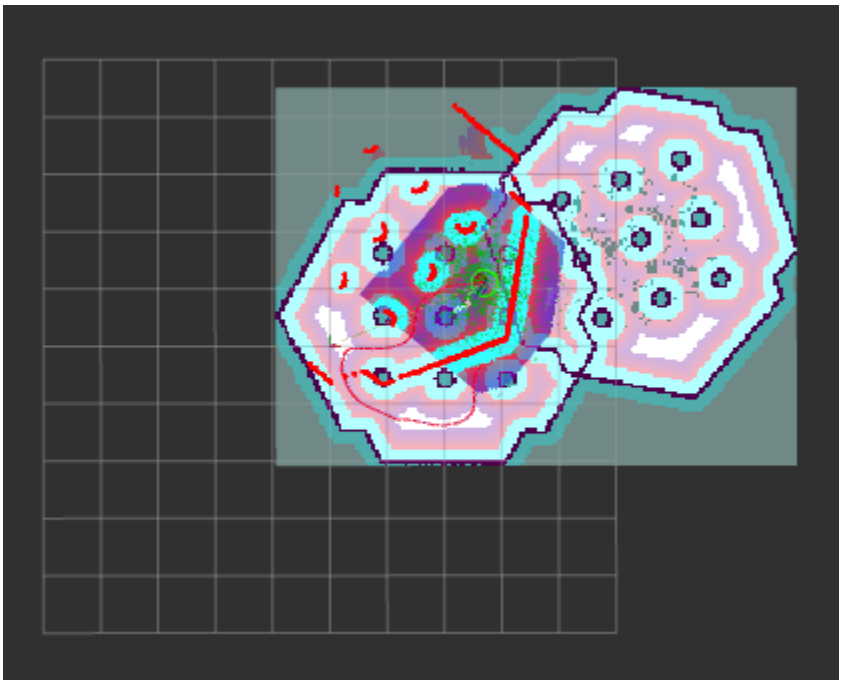
The table of six screenshots from your costmap tuning experiment in Part 2.



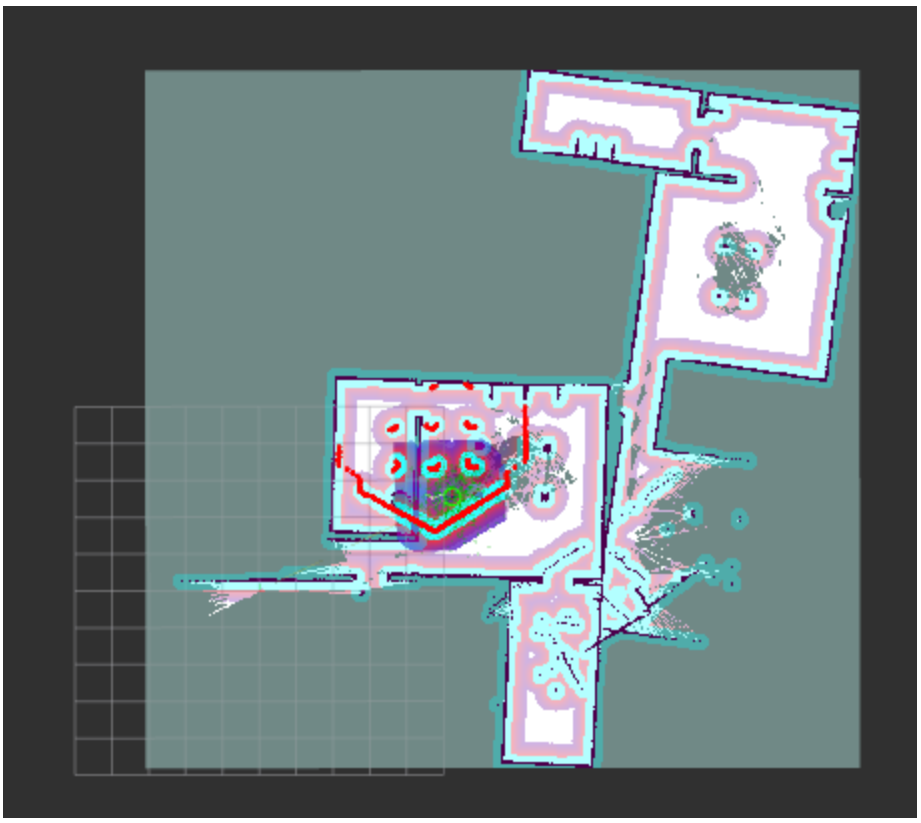
word_map_0.2_radius



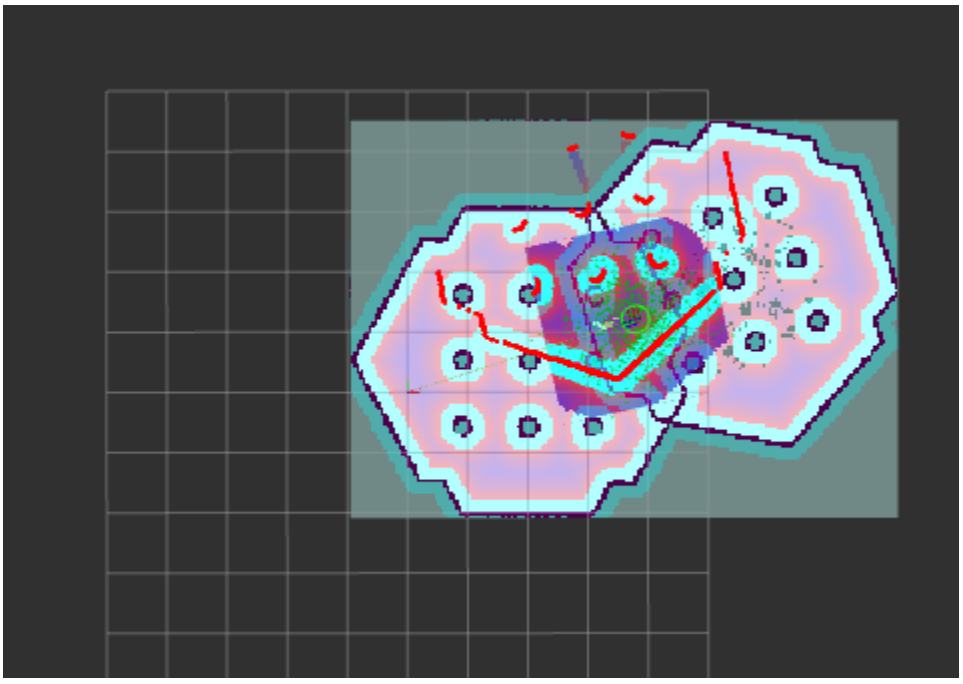
good_house_0.2_radius



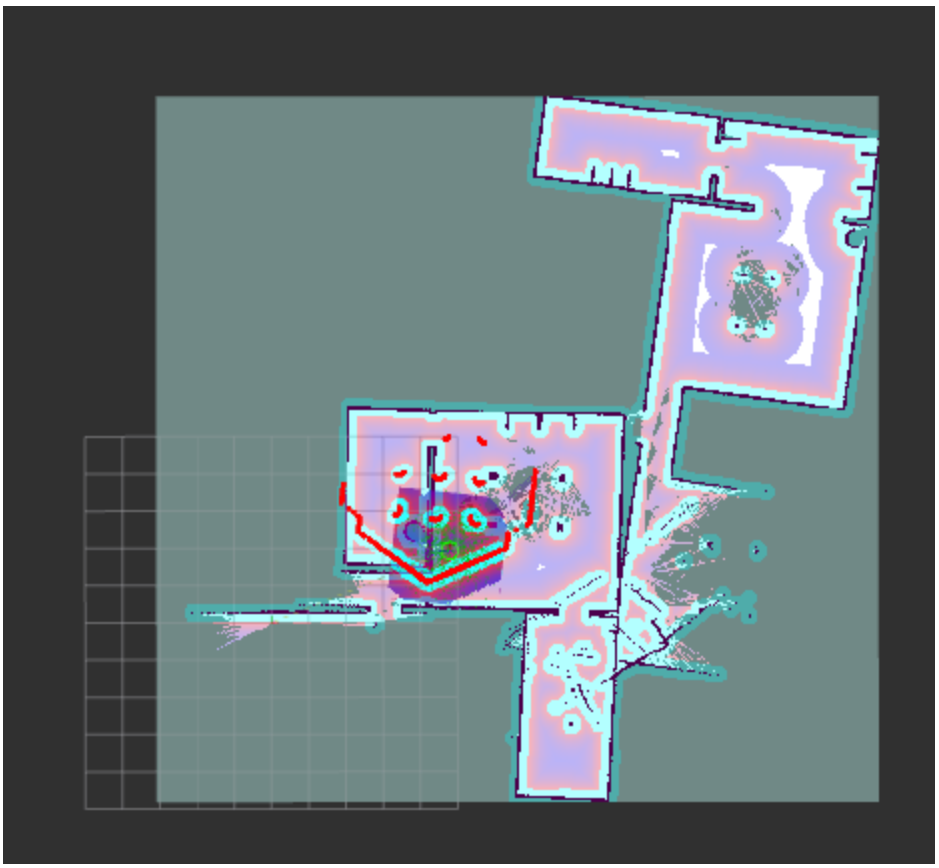
world_map_0.55_radius



house_good_0.55_radius



world_map_1.0_radius



house_good_1.0_radius

Analysis: Provide a detailed discussion of your results.

Compare and contrast the three good maps. Discuss the specific challenges each environment posed for SLAM (e.g., turtlebot3_world is small, turtlebot3_house has many identical-looking doorways).

In terms of mapping accuracy, turtlebot3_world produced the most precise representation of its environment. In contrast, turtlebot3_house achieved only about 60–75% accuracy. Due to the small scale and lack of distinctive landmarks in turtlebot3_world, SLAM struggled with reliable state estimation and consistent landmark updates. The robot was unable to differentiate between visually identical pillars, leading to map distortions, as illustrated in the resulting map. Similarly, turtlebot3_house experienced difficulties caused by its repetitive features; however, these challenges were intensified by the environment's larger size. This led to greater localization errors, evident in the robot's incorrect mapping of the lower rooms within the environment.

Compare tb3_house_good and tb3_house_bad side-by-side. Point out and label specific errors in the bad map, such as warped walls, ghost obstacles, misaligned rooms, or

incomplete areas. Explain why the aggressive driving style caused these specific SLAM failures.

From a side-by-side comparison, the tb3_house_bad map exhibits noticeable issues, including misaligned rooms and incomplete sections. The incomplete areas likely resulted from insufficient coverage during mapping - during operation, the robot primarily navigated around the perimeters of the rooms instead of moving systematically through them, as it did in the good map trial. This limited exploration prevented SLAM from fully observing and reconstructing the interior spaces. However, thorough coverage is less critical in smaller rooms, where the sensor can capture most surfaces from fewer positions.

The misaligned rooms, on the other hand, stem from the aggressive driving style, particularly the rapid turns made when transitioning between rooms. These sudden movements disrupted SLAM's ability to maintain accurate pose estimation, leading to cumulative localization errors and ultimately causing the rooms to appear offset or distorted within the map.

Discuss the mapping challenges unique to each of the three environments. How did factors like size, clutter, and room shape affect the difficulty of the task?

For turtlebot3_world, the absence of distinctive landmarks made it challenging for SLAM to accurately determine both the robot's position and the locations of mapped features. As a result, errors in state estimation led to portions of the environment being duplicated within the generated map.

For turtlebot3_house, the lack of distinctive landmarks presented challenges similar to those observed in turtlebot3_world. Moreover, the larger scale of the environment required more thorough navigation within each room to achieve complete map coverage. Due to the map's size, the robot also needed to maintain slower movement and turning speeds to reduce the risk of misaligned rooms. There was also a lot of clutter pertaining to items inside the house, so careful navigation was necessary in order not to accidentally collide the robot into walls and objects. This allowed SLAM to more accurately estimate the robot's position and maintain consistent spatial relationships between mapped features.

Analyze the differences between the high-quality and low-quality maps and explain how those differences impacted Nav2's performance.

On the high-quality map, the walls of the house are completely defined; there is only some lost data for the center of the larger rooms due to objects in that area. This resulted in the Nav2 performance being quick and smooth, where the robot was able to successfully identify all obstacles in the area and there were no "phantom" obstacles blocking its path.

On the low-quality map, there is significant information for the rooms not displayed, where several rooms are "cut-off" and essentially unknown to the robot navigating through it. This means the robot is unable to react in time to obstacles, causing collisions or even "fake" collisions where it would think an obstacle is blocking its path when there is actually no obstacle. This resulted in suboptimal pathing where the robot would take a longer route than the route in the high-quality map.

The navigation attempt using the high-quality map demonstrated optimal performance across all metrics. The global planner successfully generated a clean, direct path from the starting position in the bottom-left room to the goal location in the top-right room. The green path line visible in RViz shows an efficient route that takes advantage of the available doorways and hallways without unnecessary detours. The robot's path planning was successful because the map accurately represented all walls, doorways, and hallways with precise geometry.

The navigation attempt using the low-quality map revealed significant performance degradation and navigation failures. The global planner either struggled to find any valid path to the goal or generated a highly suboptimal path that was considerably longer than necessary. This occurred because the bad map contained distortions, ghost walls, and warped geometry that did not correspond to the actual physical environment in Gazebo. Doorways appeared narrower or completely blocked in the map, forcing the planner to either fail entirely or route around phantom obstacles.This shows that map quality is absolutely critical for autonomous navigation success. The Nav2 stack, despite being a sophisticated and well-tuned navigation system, is fundamentally limited by the quality of the map it receives.

For the high-quality maps, Nav2 stack performed well, as it was able to get an optimal path without going near obstacles and rerouting from phantom obstacles. On the other hand, low-quality maps forced the Nav2 stack to perform suboptimally, resulting in cases where the goal was not reached.

Discuss the results of your costmap tuning. Which inflation_radius worked best in which environment, and why?

The costmap tuning experiment revealed that inflation radius significantly impacts navigation safety and efficiency, with optimal values depending heavily on the environment's characteristics. With an inflation radius of 0.2 meters, both environments showed minimal obstacle inflation with very narrow blue zones around walls and objects. In the tb3_world map, this created aggressive path planning where routes came dangerously close to obstacles, leaving minimal safety margin for localization errors or motion control imperfections. The good_house map was even more problematic with this setting, as doorways appeared barely passable and narrow hallways left no room for trajectory adjustments, making the robot prone to getting stuck

or colliding with door frames. At 0.55 meters, both environments demonstrated significantly improved performance with moderate safety buffers that balanced efficiency and caution. The tb3_world paths maintained reasonable clearance while remaining direct, and the good_house navigation showed properly inflated doorways that gave the robot adequate space to maneuver through tight spaces without being overly conservative. This middle-ground setting provided the local planner sufficient room to handle minor uncertainties while still allowing efficient path planning. With a 1.0-meter inflation radius, the costmaps showed extensive blue inflation zones that created overly conservative behavior. In the tb3_world, paths became unnecessarily wide and indirect, avoiding obstacles by excessive margins that wasted time and distance. The good_house environment suffered more severely, with many doorways becoming completely impassable as the inflation made narrow openings appear blocked, forcing the global planner to find extremely circuitous alternative routes or fail entirely to find valid paths to certain rooms.

Based on these observations, the 0.55-meter inflation radius proved optimal for both environments, though for different reasons. In the open tb3_world environment, this setting provided adequate safety margins without unnecessarily constraining navigation, allowing the robot to move confidently through wide spaces while maintaining reasonable distances from walls. For the good_house environment, 0.55 meters was the sweet spot that preserved doorway passability while still providing sufficient safety buffers—smaller values risked collisions in tight spaces, while larger values made too many passages appear blocked. The important idea here is that inflation radius must be tuned based on the tightest constraints in the environment: if the map contains narrow doorways or hallways, the inflation must be small enough to keep these passages navigable, but large enough to prevent the robot from attempting dangerously close approaches to walls.

Challenges: Describe any significant technical problems your group encountered (e.g., issues with coordinate transforms, plugin registration, complex bug) and how you solved them.

We encountered technical difficulties with placing 2d pose and nav goal correctly so we asked each other to look over our maps to see if it looks about right.

Division of Labor: Include a table that clearly states who was the primary lead on each task and sub-task of the lab.

Patrick Lin	Part 1
Lina Tran	Part 2 (Navigation Results)
Lameya Mostafa	Part 2 (Comparison + Tuning)

Reflection Questions: Provide thoughtful answers to the following questions
Connecting the Dots: This lab covered the full robotics stack: mapping (SLAM) and localization and navigation (Nav2). In your own words, explain how these two components depend on each other. For example, how did a poor map from Part 1 directly cause a navigation failure in Part 2? How did the paths you made from waypoints depend on the costmap from Part 2?

The relationship between mapping, localization, and navigation forms a critical dependency chain where each component's quality directly impacts the next. SLAM creates the foundational map that serves as the static reference frame for all subsequent navigation tasks. This map becomes the ground truth that AMCL uses to localize the robot by matching current laser scan data against the stored map geometry. When SLAM produces a high-quality map with accurate wall positions, clear doorways, and proper alignment, AMCL can confidently determine the robot's position because the sensor data matches the map expectations. This accurate localization then enables Nav2's global planner to generate valid paths, knowing exactly where the robot is and where obstacles exist in the environment.

However, when SLAM produces a poor-quality map, this dependency chain breaks down catastrophically. A map with ghost walls, warped doorways, or misaligned geometry creates fundamental mismatches between what AMCL expects to see and what the robot's sensors actually detect. The particle filter spreads out rather than converging, indicating that AMCL cannot reconcile the conflicting information between the map and reality. Even if localization somehow succeeds, the global planner then treats phantom obstacles as real, either failing to find any path to the goal or generating extremely inefficient routes around non-existent walls. The local planner faces an impossible situation where the static costmap layer shows obstacles but the sensor data shows clear space, leading to erratic behavior, hesitation, and navigation failure. This demonstrates that navigation is only as good as the map it relies on—no amount of parameter tuning or sophisticated planning algorithms can compensate for fundamentally inaccurate environmental representation.

Plugin vs. Node: if you were to integrate your planners from lab 4 as a Nav2 plugin, not as a standalone ROS2 node, what are the key advantages of this approach for a complex system like Nav2? Why is it better than, for example, having your planner node publish a path to a topic that another node subscribes to?

The plugin architecture allows direct function calls and shared memory access within the same process, eliminating the overhead and latency associated with ROS2 topic communication. When Nav2's controller server needs a new plan, it can call the planner plugin directly and receive the path as a return value in microseconds, rather than publishing a request, waiting for message serialization and transport, and subscribing to a response topic which could take milliseconds. This tight integration is critical for real-time navigation, where the robot needs rapid replanning in response to dynamic obstacles or changing conditions. Thus, the planners from lab 4 would result in more accurate goal reaches, and fewer errors appearing as seen in part 2 of the lab, where the robot would often erratically move or collide into imaginary obstacles.

Group Reflection: What is one thing you learned about collaborative software development in a complex robotics project?

Documentation of our work is important because we work based on previous work from our group members. It helps when needing to debug or work backwards when software errors appear. We needed better communication as a group, as some group members did not respond on time and caused a slight delay in the progress of the lab.