Serving Mobile Robot Test Scenario

1. Purpose of project

This project aims to research and perform use cases of a serving robot in a restaurant. The robot will be tested in multiple scenarios with several route-finding algorithm to determine the best parameters and algorithms for optimizing serving time and energy.

2. Test scenario:

The environment for testing is the lobby on 2nd floor of D8 building. The dimension of the environment and furniture are indicated as in Figure 1.

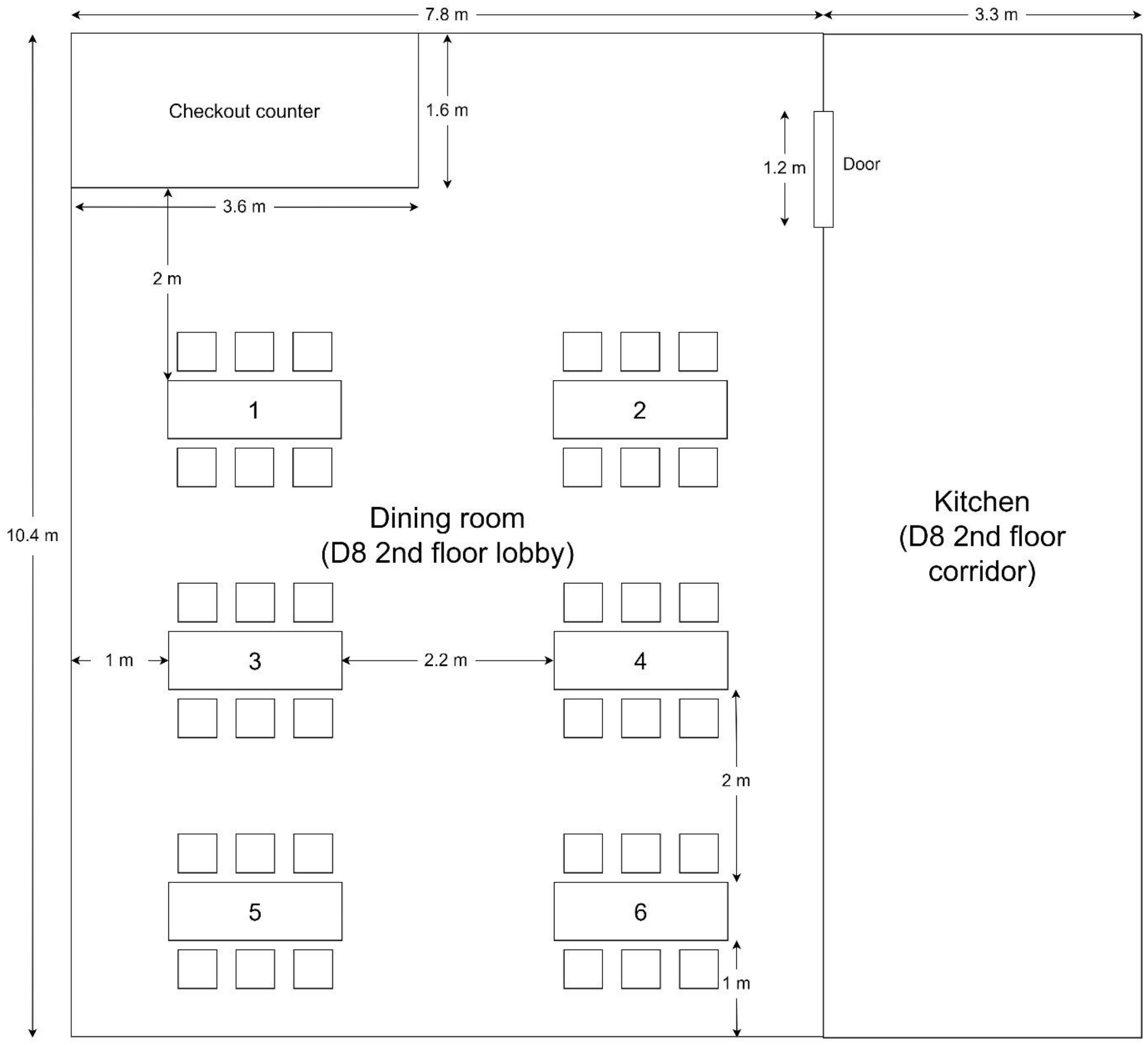

Figure 1: Testing environment

3. Use case of optimization of energy and serving time for Sevibot:

1. The kitchen staff puts the first dish on the tray and determines the corresponding table by computer (same goes for the next dishes)		
2. Robot initiates timer (2 minutes)		
3.1. Number of full tray < 3 and timer finishes counting	3.2. Number of full tray = 3	3.3 The kitchen staff presses “Start” button
4. Robot comes from the kitchen to the dining room		
5. Robot stops at the nearest table in the order list		
6. Robot initiates a timer to wait for guests to take the dishes (30s)		
7.1. The dishes are taken	7.2. The dishes are not taken and timer finishes counting	
8.1.1. There still exists table(s) need serving	8.2.1. There are not any tables need serving	
8.1.2. Robot moves and serves the nest nearest table in the order list		
9. Robot goes back to the kitchen		

Additional function:
- There is a button to command the robot to go back to the kitchen immediately
- Send Warning signal back to the kitchen staff when the robot is not able to move (so that kitchen staff can switch to manually control mode to let robot escape from “stuck” situation – a situation that the algorithm and robot themself can’t figure out the route and act weirdly, then robot will continue to auto-navigate).
Kitchen staff will receive signals from robot and be able to send signal to robot via an application interface on computer.

4. Test cases:

- Departure test:

+ Case 1: Only 1 robot’s tray is full => set goal to a single table (table 6) and wait for the timer (2 minutes) to check whether the robot departs.

+ Case 2: 3 robot’s trays are full => set goal to 3 tables (table 1, table 2, table 5) and check whether the robot departs immediately.

+ Case 3: Number of full trays < 3 and press “Start” button and check whether the robot departs immediately.

- Serving test:

+ Case 1: Set goal to 3 tables (table 1, table 2, table 5) and wait until robot goes to table 1. Then, the customer of table 1 don’t take the dish (not interact with robot) to check whether the robot goes to the next table after 30 seconds.

+ Case 2: Set goal to 3 tables (table 1, table 2, table 5) and wait until the robot has served all the tables then check whether the robot comes back to the kitchen.

+ Case 3: Set goal to a random number of tables and wait until robot goes to the dining room. Press “Go back” button and check whether the robot goes back to the kitchen.

- Obstacle avoidance test:

+ Case 1: The dining room has a few slow dynamic obstacles. Set goal to 3 random tables and check the performance of robot.

+ Case 2: The dining room has several slow dynamic obstacles. Set goal to 3 random tables and check the performance of robot.

+ Case 3: The dining room has a few fast dynamic obstacles. Set goal to 3 random tables and check the performance of robot.

+ Case 4: The dining room has several fast dynamic obstacles. Set goal to 3 random tables and check the performance of robot.

+ Case 5: Set goal to 3 random tables, when robot move in dining room, create a “stuck” situation and see if robot can send Warning signal back to computer application.

All the cases will be repeatedly tested with different Global planner algorithm (Djikstra, A*, Theta*, Hybrid-A*,...) and Local planner algorithm (Trajectory Rollout Planner, DWB Planner, Regulated Pure Pursuit,...) to compare on: time response of path planning, optimal path, ability to avoid obstacle, power consumption, pose precision.