ACME Inspection Robot

Chang-Hong Chen (117397857) , Sparsh Jaiswal (117433968), Po-Yu Huang (117684681)

Abstract - Direct testing of algorithms in the real world has huge costs and safety concerns associated with them. This is where simulation comes into picture. The agenda of this component is to create a software to automate simulation for evaluation of various algorithms that will be running on a wheeled robot.

1. Introduction

A small warehouse is required to change its layout according to seasonal demand of the products being handled in the warehouse. This warehouse has a surveillance robot that monitors certain locations in the warehouse everyday as part of the safety program of the company. There is uncertainty in how the mobile robot will perform with occasionally, slightly changing environments. This project aims to answer this question.

2. Objective

To create a simulation that evaluates the stability of robot path planning and localization by completing different routines in a partially-changing environment.

Develop a simulation software package to evaluate the robustness of the localization package of a surveillance mobile robot for the warehouse
The robot is required to consistently follow a routine every iteration with slight changes to the environment after a certain number of iterations “m”.
Robustness is be evaluated by the failure rate after “n” iteration of the simulation

3. System Implementation

The module will be developed in C++ and AIP will be incorporated for implementation. AIP allows us to maintain the quality of the work and keep a check on the resources being used for development by tracking the progress and changes closely.

Design Outline:

Three modules will be created.

Simulation (Will be developed):
Simulation module controls everything regarding the simulation environment.

Robot (Provided by customer):
Robot module controls the behavior of a single robot

Fleet management (Will be developed):
Fleet management is the central control of the whole warehouse monitoring process. It runs a state machine and controls the main workflow.

4. Assumptions

1. Once an iteration starts, the world remains static meaning all the scenes will be fixed at its current position. The only thing that will be moving is the robot. This assumption is to simplify the problem by avoiding dealing with dynamic obstacles.
2. Movable scene in the environment will change its position for every m iteration, as a way to simulate the seasonal changes in the layout of the

warehouse according to the product being handled in that season.

3. A routine will be given to the robot to complete for every iteration.
4. The movable region for movable scenes is restricted. The movable scenes will not block or occupy the waypoints and the origin. And the change of position will be picked randomly in the movable region before iteration starts.

5. Definitions

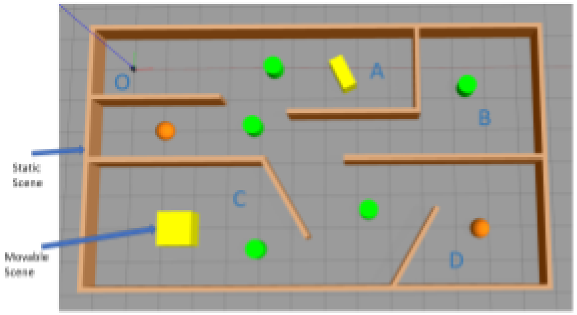
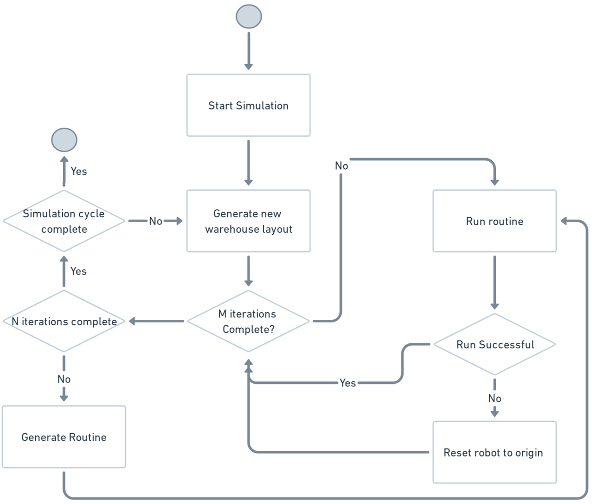


Fig 1. Environment example

- **Routine:**
 A set of waypoints the robot has to complete in a given order. Ex: **A→B→C→D**
- **Iteration:** A single run of the simulation that involves a routine in a fixed environment.
- **Static scene:** Non-movable obstacles in an environment throughout every iteration, including Static Racks, Walls (The boundary of the environment), Pillars, etc.
- **Movable scene:** Obstacles that may change its position between iterations, including Movable Racks.
- **Environment:** A simulation world that consists of movable and static scenes.
- **Origin:** A fixed location in the environment which is the reference point for every scene and tf. **O** in fig 1.
- **Routine Success:** The event of when the robot is able to complete the routine in a predefined amount of time.
- **Routine Failure:** The event when the robot is unable to complete its routine in the given amount of time.

6. Flow Diagram



7. Risk:

- Communication blackout between packages
- Inaccurate robot model

8. Deliverables:

1. Evaluate the success rate with and without remapping after every change.
2. Simulation setup for regression testing

9. References

[1] robot_localization: http://docs.ros.org/en/noetic/api/robot_localization/html/index.html

[2] slam_toolbox: http://wiki.ros.org/slam_toolbox

[3] Nav2: https://navigation.ros.org/