Warehouse Robots for Acme Robotics
Project Code: 808XFP

Introduction

The idea is related to collaborative robots or "cobots". The mobile robots are designed to help human workers perform diverse tasks in warehouse environments and may also have the capability of acting as a mobile storage bin for picked orders. They also have the capability to navigate autonomously using various object detection, path planning, and perception techniques. We are proposing to add these Autonomous Mobile Robots (AMRs) to the portfolio of Acme Robotics

Fig 1. A warehouse robot
Source:Shutterstock.com

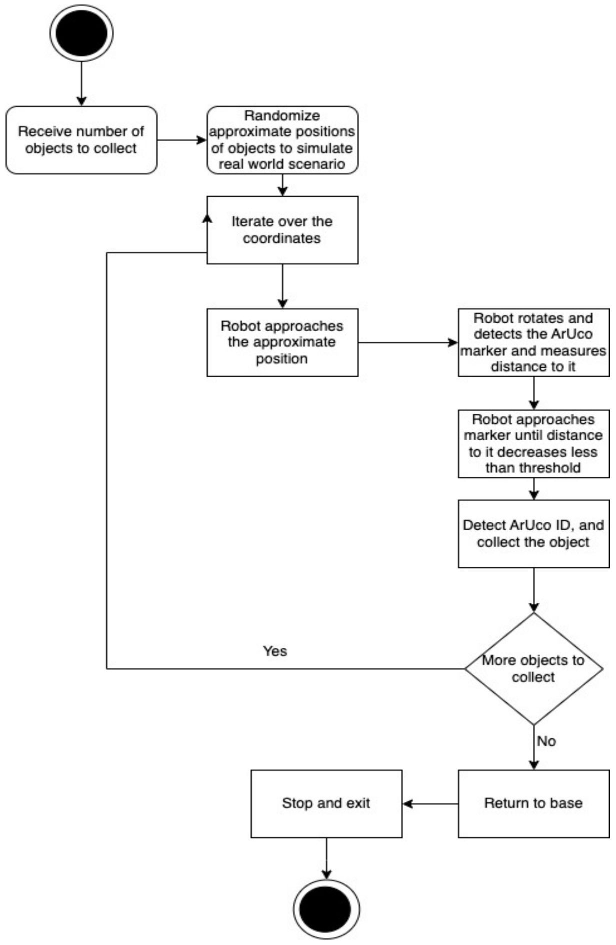
Trends to make this B-plan a success:

Multi-cloud-based Warehouse Management - Large warehouses like amazon deploy multi-cloud-based Warehouse Management Solutions (WMS), and these AMRs are easily integrable and help in inventory management.

Mobile Robots as a Service (RaaS) – A rental service that allows warehouses to lease the AMRs, decreasing capital investment.

Similar other trends include Automated Storage and Retrieval Systems, IoT, Big Data and Analytics, Voice Picking, Wearables, Smarter Layouts, etc. Adding to this, the retail industry is stuck in a strange situation; the sales are increasing, and store traffic returning to normal but the overall US economy is feeling the heat of rising inflation and labor shortage. Retailers across the country are already preparing for an anticipated recession and devising ways to minimize costs and find efficiencies through operations.

Architecture


Receive number of objects to collect
Randomize approximate positions of objects to simulate real world scenario
Iterate over the coordinates
Robot approaches the approximate position
Robot rotates and detects the ArUco marker and measures distance to it
Robot approaches marker until distance to it decreases less than threshold
Detect ArUco ID, and collect the object
More objects to collect
Yes
No
Return to base
Stop and exit

Algorithms

1. Monte Carlo Localization: Used by the ROS MoveBase package to localize the robot in the map
2. A* algorithm: Used by ROS MoveBase to plan paths
3. Perspective Projection: Used by ROS ArUco detector package.

Design and Development

For development purposes, we will be using C++ 17 along with ROS 1 (Noetic). AIP will be used and frequent commits with proper commit messages will be made during the development. The pair programming will be executed considering the Driver (Adithya), Navigator (Rishabh) and Design Keeper (Divyansh). Cpplint and cppcheck will be used to ensure the code quality. Code coverage will be done using Coveralls. Open source MIT license will be added to the project. Version control will be taken care of and release will be made once the development is finished. Object Oriented Programming will be used in the code.

Assumptions:

- Gazebo world is chosen to represent a warehouse.
- Turtlebot is assumed to be a warehouse robot.
- No dynamic obstacles are considered.

Deliverables:

The software stack will be written in C++ with ROS packages and simulated in Gazebo on the turtle bot simulator platform. The robot will receive random approximate points where objects need to be collected following which it will traverse a path to the coordinates and search for objects to be collected using ArUco tags. Once detected, the robot will measure the distance between the object and itself, and approach the object until the distance is smaller than the threshold and finally collect it. This will be done when all objects have been collected. The project on the simulation world will have various assumptions and helps us understand how our robot interacts with the real world and the factors we need to consider before manufacturing an actual product. Along with the above deliverables, we will have an initial UML design, budget analysis, and backlogs.

Risk and Mitigation

Automation should be done considering growth in the future. For example, enough space should be available to accommodate more robots. Plan the whole process before automating. Don't always aim for complete automation, sometimes partial automation can give a better ROI as compared to complete automation. Before automating any process, study the environment thoroughly. For example, what is the temperature in the environment, and will the robots be able to accommodate it? Also, our robot has not been optimized for a multi robot situation, in case we want a swarm of robots, this might not be the best solution. For the best results from the algorithm and AMRs the surface friction needs to be maintained and avoided from getting oily or dirty.

Following are the steps that can be a part of the mitigation process:

In case of a fire alarm, the robot should stop moving ensuring it is not blocking any emergency exits. In case of a fire, the whole system should shut down automatically. There should be multiple sprinklers in the environment to put out the fire. Robots should have temperature sensors that send out an alert in case of abnormal temperatures. Automatic emergency braking should be present in the robots to avoid collisions. In addition to the above strategies, the robots should be insured to minimize the loss in case of any hazardous situation.

References: (ROS REPS, libraries)

- https://index.ros.org/r/turtlebot3/
- http://wiki.ros.org/move_base
- http://wiki.ros.org/gmapping
- http://wiki.ros.org/tf
- http://wiki.ros.org/actionlib
- http://wiki.ros.org/geometry_msgs
- http://wiki.ros.org/std_msgs
- http://wiki.ros.org/aruco_detect