Autonomous Picking Bot

ENPM808X - Final Project Proposal

Nitesh Jha
Master of Engineering in Robotics
University of Maryland - College Park
niteshj@umd.edu

Tanuj Thakkar
Master of Engineering in Robotics
University of Maryland - College Park
tanuj@umd.edu

I. Software Plan

A. Overview

B. Deliverables

- C++ based ROS Package to detect, identify and pick/-place cargo in a warehouse environment
- UML Diagrams
- Developer-level Documentation of the C++ based ROS package
- Code-base following Google styling guidelines for C++
- GithubCI for continuous integration and Coveralls for code coverage

C. Definitions and Acronyms

- Staging Area: The area where cargo to be stored in the warehouse is placed after unloading from the trailer
- Cargo Bay: The area where cargo is stored in the warehouse
- ArUco Tags: They are binary square fiducial markers that can be used for pose estimation

D. Methodology

1) Task Allocation: The operation begins with allocation of tasks, which will be based on a queue of cargo to be picked by the robot. This module emulates the management of incoming cargo in the warehouse.

2) Navigation: In the warehouse environment, the location of the staging areas, and cargo bays is known. The robot will navigate to respective target location as per the task being executed using a simple path planner and a low-level controller.

3) Search & Identification: All the cargo in the warehouse environment is assumed to have ArUco Tags on them. The cargo will be randomly spawned in the staging area and has to be transferred to a cargo bay as per the assigned task. The known location of keypoints of the staging area utilized to search for a particular cargo with respect to the task. The robot will visit these keypoints of the staging area, search and identify the cargo of interest to execute the task.

4) Pick and Place: On successful identification of the cargo, the pose of the cargo estimated in camera frame is then transformed to manipulator frame. It is then picked up by the manipulator. They are then carried to the respective cargo bays and placed.

II. Project Organization

A. Design

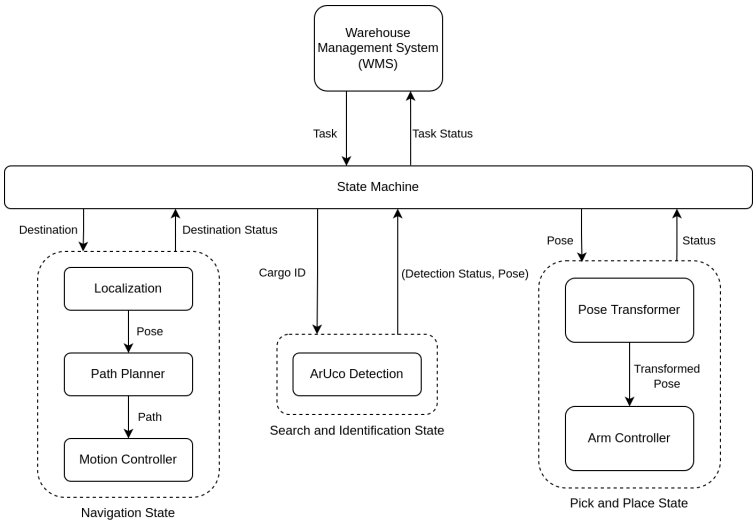

Fig. 1: System Design

B. Organizational structure

1) Phase 1
   - Tanuj Thakkar
   - Nitesh Jha
2) Phase 2
   - Tanuj Thakkar (Driver)
   - Nitesh Jha (Navigator)
3) Phase 3
   - Nitesh Jha (Driver)
   - Tanuj Thakkar (Navigator)

III. Managerial Process

A. Assumptions, dependencies, and constraints

- The warehouse environment is fully explored and the occupancy grid is available for navigation
- All cargo have ArUco tags on all sides
- Size of the ArUco Tags is known
- Staging area and cargo bay locations are known.
- There are no dynamic obstacles in the environment.

B. Monitoring and interfacing mechanisms

Git will be used for version control of the module codebase. For continuous integration, GitHubCI will be employed along with Coveralls for tracking and analysing code coverage.

IV. Technical Process

A. Tools and Techniques

- Ubuntu 20.04 (LTS)
- C++ 14 or higher
- CMake 3.10.0 or higher
- Git
- GithubCI
- Coveralls
- ROS2 Humble Hawksbill
- Gazebo 9
- RViz

B. Dependencies

- OpenCV 4.6.0
- Eigen 3.4.0
- Boost 1.71.
- Gtest
- GMock
- ROStest
- Aruco Detect
- MoveBase
- MoveIt

C. Software Documentation

The developer level documentatation of the package will be provided using Doxygen.

References