Introduction to Robotics - Helpful Robot Lab

Nicholas Jones

May 2, 2020

1 Introduction

This proposal outlines the design for a grocery shopping robot. Its purpose is to semi-autonomously navigate through a grocery store in order to collect items for a remotely located user. This allows for the curbside pickup of groceries by high-risk individuals in a pandemic situation, without the requirement of human intervention.

2 Requirements and Constraints

2.1 Requirements

The robot must:

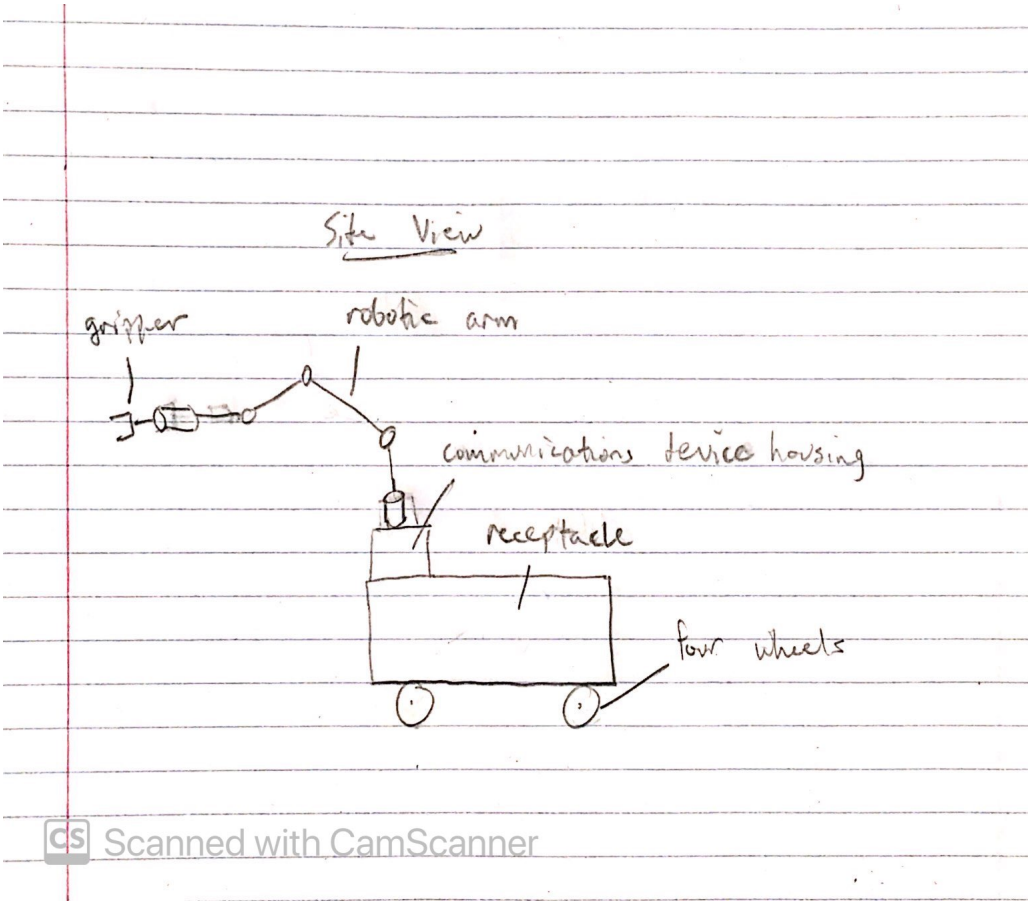
1. dynamically avoid obstacles during operation
2. plan a general path that intersects the locations of all requested items, optimizing for path length when possible
3. use computer vision to locate specific items once in their general area
4. be able to manipulate a variety of food items
5. be able to place items in its receptacle without damaging them
6. receive user grocery lists remotely through wifi connection to a user application
7. localize within the store and surrounding parking lot area
8. include a clearly marked and easily accessible emergency stop button which, when pressed, commands all joints to hold their positions or cuts power to all joints

2.2 Constraints

The robot must:

1. be able to carry up to 50 lbs of groceries in its main receptacle
2. be able to carry up to 4 cubic feet of groceries in its main receptacle (a slightly smaller volume that a normal grocery cart)
3. occupy envelope dimensions not exceeding 2 ft wide x 4 ft long x 3.5 ft tall when in a non-operational state
4. have sufficient battery life to operate for at least 4 hours continuously (designed for use during peak times, can be charged throughout the day)

3 Overall Design Depiction



4 Subsystems

1. Mobility/Platform - This subsystem will encompass the physical platform of the robot and it's driving capabilities. It will satisfy constraints 1 through 4 above. This subsystem will also deal with the safety features of the robot, fulfilling requirement 8 above.

2. Path Planning/Obstacle Avoidance - This subsystem will deal with the robot's predetermined trajectories as well as minor adjustments during their execution to avoid dynamic obstacles (like people). Requirements 1, 2, and 3 fall under this subsystem.

3. Localization - This subsystem will use data from the mobility and path planning subsystems to inform the robot's current location, which will be available to the system administrators throughout operation. Requirement 7 above pertains to this subsystem.

4. Manipulation - This subsystem will consist mainly of a robotic arm with a gripper attachment

on the end-effector, as well as a prismatic joint that adjusts the height of the arm. It will satisfy requirements 3, 4, and 5.

5. Communications - This subsystem will facilitate networking between each other subsystem and will also be responsible for requirement 6 above.

5 Trade Study 1 - Robotic Arm

Option 1: Hebi Robotics 5 DOF wtih X-Series actuators: https://docs.hebi.us/resources/kits/datasheets/x-series/A-2085-05_Datasheet.pdf
Option 2: Kuka KR Cybertech Nano: https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/kr-cybertech-nano

The Hebi arm is easily programmed, lightweight, and easily reconfigured. However, it would also be less precise in its operation and has a lower payload. The Kuka arm is meant for more industrial applications, so it is much heavier than the Hebi arm, has a higher payload, and is more precise. Once assembled, it cannot be reconfigured (in terms of rearranging joints or links) and its rigidity during operation would make it a less safe option. I would choose to use option 1 since safety, flexibility, and weight are especially important when designing mobile robots.

6 Trade Study 2 - Central Computer

Option 1: Raspberry Pi: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
Option 2: Nvidia Jetson TX2: https://developer.nvidia.com/embedded/jetson-tx2-developer-kit

Either option could be sufficient to interface between the various subsystems, however the Jetson contains a bit more graphics processing power. Given that the robot needs to use computer vision in real time to locate specific grocery items and plan arm trajectories to pick them up, the I would choose to use the Jetson over the Raspberry Pi.

7 User Interface

The robot will effectively have two types of operators: the system administrator (someone stationed at the grocery store whose responsibility is to monitor the operation of all individual robots) and the users (store customers who will use the app to order groceries and then physically unload them directly outside the store). The only interface component physically located on the robot will be the emergency stop button, which the operator can configure to either hold the robot's current pose or cut the power. The emergency stop procedure that is chosen should be subject to a deeper analysis - cutting the power could case a heavy component such as the arm to fall and strike someone, while holding the pose could potentially pin someone in a dangerous position, and it is unclear which outcome is "worse". There may be a hybrid option where the pose is held but with very low resistance to physical intervention. The emergency stop feature and failure state is crucial to the robot's operation and must be robust, because there is a large risk of accidental contact with humans.

8 Implementation

I chose to implement the path planning algorithm and a simple GUI to visualize and test it.

The code is located here:
https://drive.google.com/file/d/1VmV_BLItYDa4lo5NtiDB5BsCEp7QZk_l/view?usp=sharing

A video of the implementation being tested is located here:
https://youtu.be/_tUQfAZnw54

The program simulates a grocery store environment. Nodes 11 and 12 represent the entrance and exit, respectively, and all other numbered nodes represent food items spread throughout the store. The search algorithm is a naive approach to the traveling salesman problem. It begins by taking a list of nodes to traverse as input and determining all possible permutations of them, inserting node 11 at the beginning of the list and appending node 12 to the end. Then, for each permutation, it performs Dijkstra's search on each pair of adjacent nodes and concatenates the results into one path. Finally, it returns the shortest path (having kept track of path length when calculating a path from each permutation). This approach runs in O(n!) time, so it is quite inefficient and only works for a maximum of about 7 or 8 nodes (not including 11 and 12). For this reason, when 8 or more input nodes are given, the program bypasses the permutation approach and just calculates a non-optimal path with respect to path length. The user interface is relatively self-explanatory, and it should be able to handle any type of input without crashing.

The validation process consisted of testing several input edge cases, such as zero nodes, a large number of nodes, repeated nodes, and illegal (non-integer or out of range) inputs. For a real-world implementation of this robot, a more complicated dynamic programming approach would reduce the time complexity of the search algorithm somewhat. However, this implementation does meet requirement 2 to plan a path intersecting the locations of all the requested items, optimizing for path length for small inputs (i.e. when possible).