Autonomous Package Delivery Bot

Sajidur Rahman	2011B5A7496G
Shaik Afroze	2013A7PS054G
Sarangapani Manojchandra	2013A7PS030G

1 OBJECTIVE

The project dealt with localization of a bot and a user and autonomous navigation of the bot within the boundaries of an indoor environment. We aimed to build an autonomous robot that performs the task of indoor package delivery. The user will be able to call the bot to his location with the press of a button. An extended application of such a robot will be to make home speaker systems highly portable by fixing them to the bot and making the bot follow the user around the house. The autonomy of the robot required a high degree of automated navigation. The ability to successfully localize the robot and the user is a key requirement of such a robot.

2 MATERIALS AND MODULES USED

This project comprised of the following hardware and software components:

Hardware:

a. Bot Chasis with 4 motors and wheels
b. Motor Driver
c. RaspberryPi 2/3
d. WiPi Wifi Module for RaspberryPi 2
e. Microsoft Kinect or any other camera device
f. Breadboard
g. Connecting Wires
h. 6 x AA Batteries for motors
i. 5v power supply for RaspberryPi and Motor Driver (preferably from a Power Bank)

Software:

a. Python
b. OpenCV
c. Numpy
d. Libfreenect (Kinect library for python)

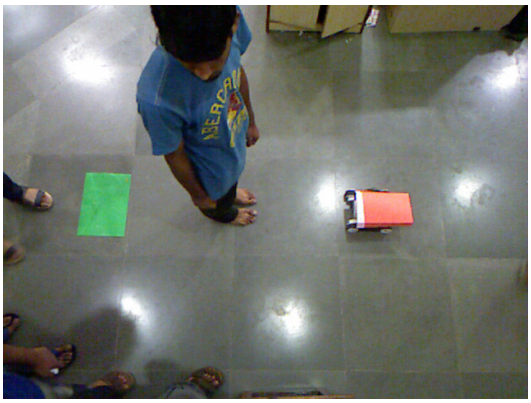
3 METHODOLOGY

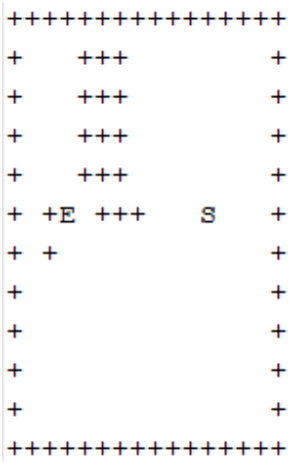
We examined the objective properly and discussed several approaches to achieve the objective in an optimized way. Our approach can be divided into the following salient points:

1. Take a snapshot of the entire floor space from a camera mounted on the ceiling of the room and feed it to a laptop running the openCV script.
2. The python script scans for contours using openCV's inbuilt function `findContours`. Take the list of contours and sort it by the area of the contour.
3. Draw bounding boxes around the contours using openCV's function boundingRect and save the coordinates of the box
4. For each area enclosed by the bounding box, find the area which has the highest average red value. Label that area as the 'source'. Similarly find the area with the highest average blue value and label that area as the 'destination'
5. Feed this information to a function that converts the image and the boundingBoxes list to a map where obstacles are denoted by 'X', the source by 'S' and Destination by 'D'.
6. Feed this map to a maze solver program that runs a Breath First Search algorithm on the map and gives the shortest path to traverse from S to D.
7. Translate this path information to instructions for robot, example move forward, backward, etc. This instruction is communicated with the Raspberrypi fitted In the robot through socket programming and the server program running in RasPi converts the instruction into GPIO pin output instructions and consequently runs the bot.

A careful and patient effort was given for properly calibrating the distance and the sleep time for the instructions in the microcontroller. To overcome the problem, we used a dynamic approach where the snapshot was taken after every second and the path recalculated and the instructions retransmitted to the bot. This gave a better result and accuracy. It is advised that the encoder pins of the motor are used in future, which would give finer results. The encoder pins give information about the exact angle by which the wheels have rotated during an interval, which can be used to calculate the exact distance moved by the robot and the angle of rotation as well.

The images below give an outline of the methodology:





Turn Left
Go Forward
Turn Right
Go Forward x 5
Turn Right
Go Forward

[Socket Programming]

GPIO Instructions in RaspberryPi

It is to be noted that all processing took place in the code running in a laptop. The script had a code for a client socket that sends the instructions to a server program running in the RasPi.

The code of the project has been well commented and can be understood from the first read itself. The entire code can be broadly divided into 3 sections:

1. Image Processing using OpenCV
2. Maze Solver (BFS algorithm)
3. Socket Programming

Out of the 3 sections, our work was mostly devoted towards the first section. If a correct representation of the contours is achieved then the rest of the code simply finds the shortest path and feeds the instruction to the RaspberryPi server program.