Robot Navigation and Path Planning Using Image Processing

Heethesh Vhavle Naresh
Electronics and Communication Engineering
B.M.S. College of Engineering
Bangalore, India
heetheshvn@yahoo.com

***Abstract*—There has been an increasing necessity for autonomous robots which can act on their own without human intervention. One of the major challenges faced is the navigation of these autonomous robots. This paper presents a technique of robot navigation solely using image processing for path planning and navigation using feedback. Navigation by this technique would not be a function of any sensors on the robot. It only requires an overhead camera with the arena and the robot in its field of view. Live feed from the overhead camera is sent to a computer where the image is analyzed and a path is computed to reach the goal. The camera monitors the arena constantly and ensures that the robot navigates the path without any deviations and avoiding obstacles.**

Keywords—navigation; path planning; image processing; autonomous robots; computer vision; localisation and mapping; position control; computer aided instruction

I. INTRODUCTION

Today, autonomous robotic systems are widely used in industries in performing tasks such as manufacturing and packaging. However, for transportation, such robots are not practical. There is currently a shift in the set of applications away from the industrial setting towards office and domestic application. Applications such as in health care facilities and offices in assisting in tasks such as fetch-and-carry, delivery and mobility aids. In industries, it is used in transportation in large warehouses and in logistics.

This paper covers aspects of navigation of these robots in an indoor environment. Image Processing is used to analyze the environment, compute a suitable path, and provide instructions and feedback to the robot to help navigate.

In all the, above mentioned, tasks that the robot has to carry, three fundamental questions arise for the system: "Where am I?", "Where am I going?", and "How do I get there". The first two questions deal with localization of the robot and the mapping of the environment into a digital form to facilitate path planning. A camera placed over the environment, with robot in it, captures live images, which are then sent to a computer for further processing. The third question deals with path planning. This stage includes choosing the most efficient path for the robot and converting the computed path data into a set of commands that the robot can understand. The final stage is communication with the robot, which is achieved wirelessly using ZigBee Protocol (IEEE 802.15.4). This stage also ensures that the robot navigates along the computed path without any deviation and is vulnerable to changes in the environment.

II. CARETAKER ROBOT

This paper is in context of a project based on a Caretaker Robot, used in health care facilities as well as in homes and various other domestic applications.

We live in a world that is fast moving towards longer life expectancies and nuclear families giving raise to care centers for the aging population. This creates the need for a large workforce to care for those who can no longer take care of themselves. Advances in robotics and computer vision have enabled us to build machines that interact with humans naturally, require no training to operate and carry out simple tasks. Thus, caretaking is a need that can be satisfied by robots, specifically designed to fulfill requests for simple services in a timely manner.

The theme consists of an arena representing a floor of a hospital having three zones: Patient zone, Service Zone and a Corridor connecting the two zones. The robot is informed of the patients' requests via a computer that processes the images from a camera, as shown in *Fig. 2* and it autonomously seeks the provisions to be picked up and delivered to the corresponding patients. Python with OpenCV module is used for image processing. Red marker indicates medicine; yellow, thermometer; blue, water. The robot can be placed in any of the three start positions. The green walls in the corridor can be present anywhere in the eight possible locations, whereas the beds, tables and partitions are fixed. *Fig. 1* below shows the arena configuration. The movement of the robot is restricted to only four directions.

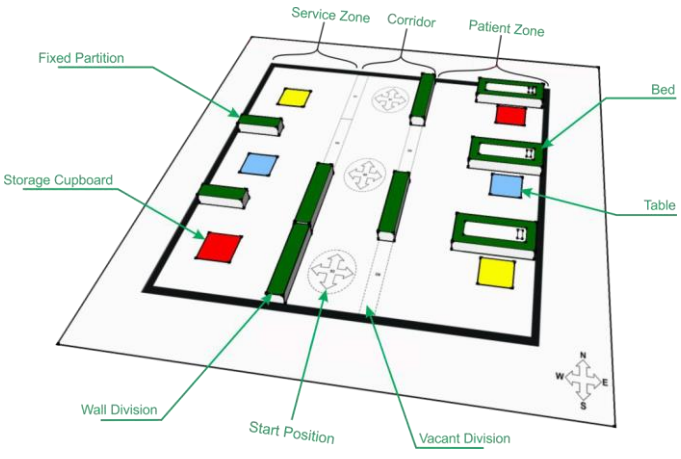

Figure 1: The arena configuration showing the various zones and location of pick-up/delivery points and the obstacles.

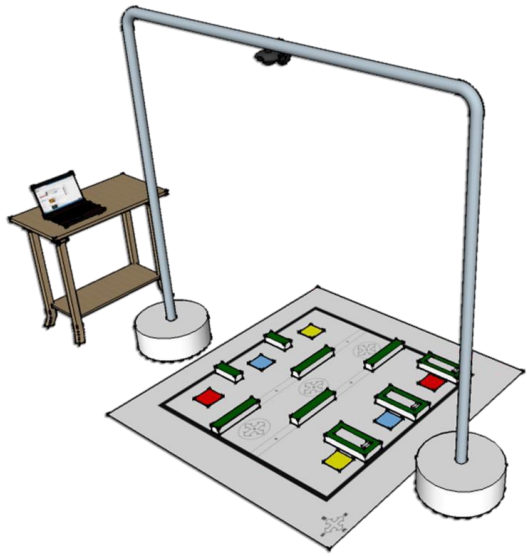

Figure 2: The arena setup with the overhead camera.

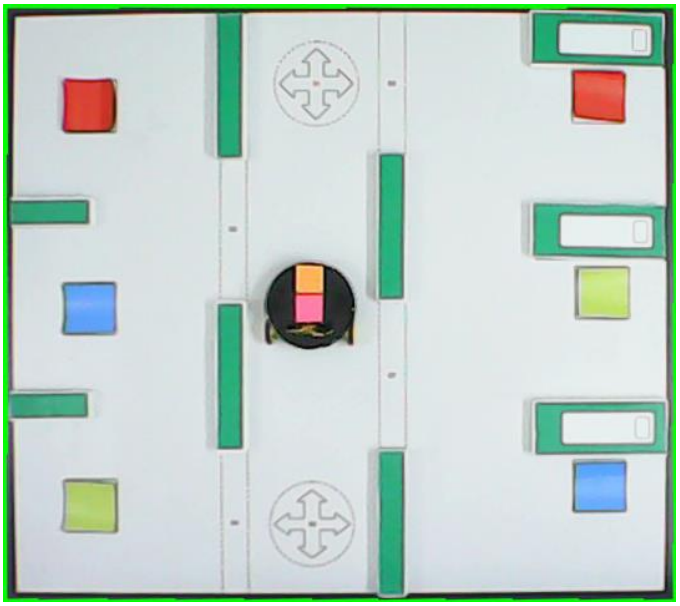

Figure 3: The input image captured by the webcam (perspective corrected).

III. Localisation and Mapping

The goal for an autonomous robot system is to be able to construct a map or floor plan and to localize itself in it. Mapping is done in order to know the start position, pick-up and delivery locations, and the position of obstacles. Localization assigns the start position of the robot in the arena and is helpful to know the position of the robot during its course at any given time. A map is created with a suitable grid size where each grid is assigned a set of co-ordinates.

A. Localisation of the Robot in the Environment

The first step is to determine the position of the starting point of the robot in the arena. In this given arena, the robot has three starting positions in the corridor region. The position is determined by finding the average RGB values over a set of pixels in each of the three start positions. By comparing these average values with a threshold value, the position of the robot can be found.

To find the orientation of the robot, two different colored markers (orange and pink) are placed on the robot. These colors are filtered and their respective centroids and found. By finding the slope of the line joining these two centroids, the direction where the robot is headed is known.

B. Mapping of the Environment

Mapping of the environment is necessary for path planning. The various positions and the type of the requested provisions, which are represented by the different colors, as well as the positions of the obstacles in the arena are found out. This is done by comparing the RGB values of pixels in the known positions of provisions and obstacles and determining the type and number of provisions and also whether an obstacle is present or not.

This process also includes creating traversable paths in selected regions, represented by white color as shown in *Fig. 4*. The gray region represents the areas where the robot cannot travel in order to avoid probable collision with the obstacles. *Fig. 3* shows the input image from the overhead webcam and *Fig. 4*, the mapped arena generated by the program.

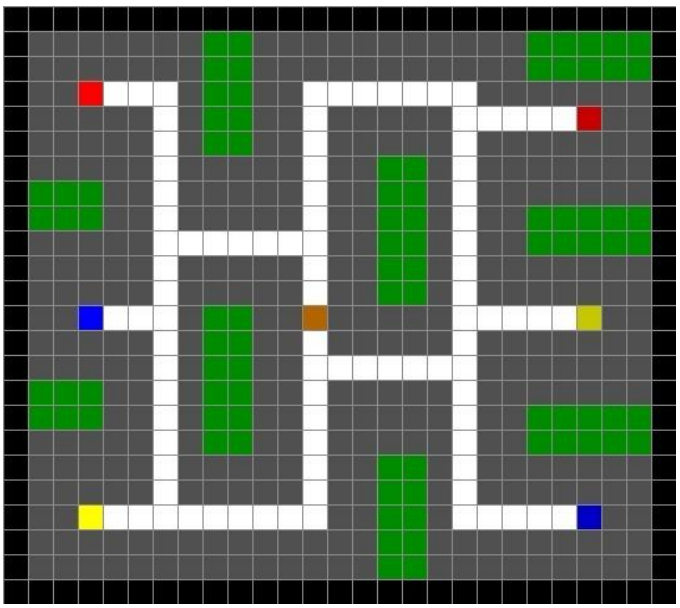

Figure 4: The arena map generated by the program.

IV. Path Planning

The next stage, after localization and mapping of the arena, is computing a path for the robot to navigate through the arena. The task includes a maximum of six path combinations, that is, three paths for pick-up and three paths for delivery. The overall final shortest path (master path) is the correct order for a combination of these six individual paths, which would be the most time efficient and shortest. Path planning stage here not only includes computing a path, but also selection of the best overall path to be followed.

The positions and the colors of the requested provisions and the provisions available in the storage area are matched and their coordinates are known. The entire mapped arena is converted to a 2D array where each element or node of the array represents a grid in the arena map. Each element is assigned a value as: 1 – Traversable cell; 0 – Obstacle / Non-traversable path; 2 – Pickup / Delivery location.

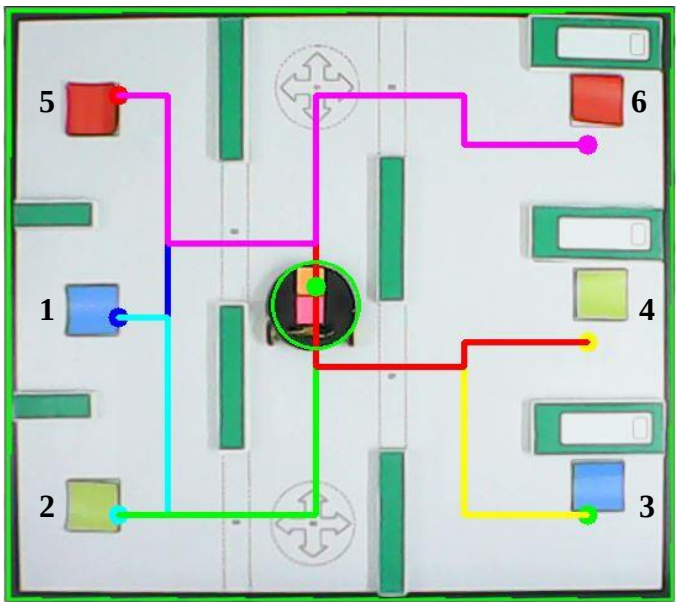

Figure 5: Path computed by the program. The colored dots mark the end of each individual path (order of paths followed - Blue, Cyan, Green, Yellow, Red, Purple).

The algorithm used for path planning is A* algorithm. A* uses a best-first search and finds a least-cost path from a given initial node to one goal node. As A* traverses the graph, it builds up a tree of partial paths. The leaves of this tree are stored in a priority queue (binary heap) that orders the leaf nodes by a cost function, which combines a heuristic estimate of the cost to reach a goal and the distance traveled from the initial node. Specifically, the cost function is

$$f(n) = g(n) + h(n) \quad (1)$$

Here, $g(n)$ is the known cost of getting from the initial node to n; this value is tracked by the algorithm. $h(n)$ is a heuristic estimate of the cost to get from n to any goal node. Manhattan distance is used for computing the heuristic value here. This heap is known as the open list.

At first, the start node is added to the open list. The node with the lowest f value is popped and its adjacent nodes are found out. The current node is set as the parent node and is added to another list known as the closed list. Then the adjacent nodes are processed to get the lowest f value and added to the open list. The algorithm continues in the same manner until the end node is reached. Then, starting with the end node, the parent node of all the nodes is back traced until the start node is encountered. The reverse order of this set of nodes gives the shortest individual path between two points.

All possible individual paths (from one pickup to delivery) are computed and the path lengths are found. Then a selection algorithm determines the order of the pickup and delivery paths for which the combination of these six paths (master path) would be the most efficient and shortest of all. If two master paths have the same length then the one with the least number of turns is selected.

Fig. 5 shows the master path computed by the program, where each color represents an individual path. The order of the paths is: Blue, Cyan, Green, Yellow, Red, Purple. At the end of each individual path a colored dot is drawn, to help identify the order of paths / position of the robot at each stage. According the *Fig. 5*, the robot first travels to the blue marker on the left to pick up 'Water', then it picks up 'Thermometer' (yellow marker on the left) and then delivers the 'Water' first and the 'Thermometer'. It then comes back to pick up the 'Medicine' and finally delivers the last remaining provision. The purple dot represents the position of the robot at the end of the run.

V. Communication with the Robot

The master path computed from the path planning algorithms yields a path in the form of a list of co-ordinates. This lists needs to be converted into a list of motion commands, which can be sent to the robot. Based on the differences between two successive set of co-ordinates, an algorithm is designed which will encode the path into a set of commands. This list of commands also includes the information for switching on/off the RGB LEDs at the right time to indicate the provision, which the robot is carrying. All repeating commands such as multiple forward/backward movements and combined, where each amount of distance is assigned a particular code. Similarly, turns and LED information are also assigned unique codes.

ZigBee Protocol (IEEE 802.15.4) is used for communication between the computer and the robot, which is implemented using *XBee* Modules mounted at both the ends. *XBee* uses line-of-sight, wireless serial communication over 2.4 GHz frequency band. *Fig. 6* shows a block diagram of the communication between the robot and the computer.

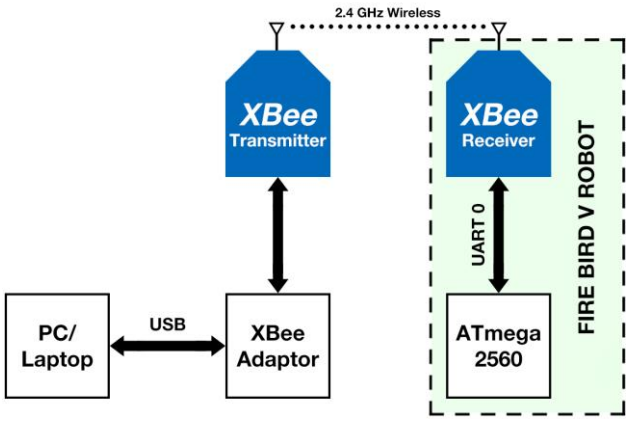

Figure 6: Block diagram representing XBee communication.

VI. Navigation with Feedback

The entire navigation information, that is the path data and LED status to indicate the provisions that are being delivered, is computed prior to the start of the actual run itself and the commands to be sent are stored in the form of a list. The commands are sent one after the other to the robot. After every command sent to the robot, the computer waits for a response back from the robot in order to send the next instruction.

Position encoders present on the robot are used to provide internal feedback to the robot to travel accurate distances and take accurate turns. Pulse width modulation is used to control the velocity of the motors of the robot. However, while the robot is navigation through the course, there will be some inaccuracy when the robot takes turns. In addition, there would be some variation in the distance travelled by the robot. Such errors are cumulative and the error at the end of the run would

be very significant. Therefore, such errors need to be corrected after every turn the robot takes and the position of the robot needs to be tracked to avoid deviation from the actual path. A feedback algorithm checks the orientation of the robot after every turn and sends back a correction angle to the robot, in case of any error present. At positions, where the robot is very close to the obstacles or near the boundaries of the arena, the algorithm checks the position of the robot and sends instructions to correct the offsets and maintain a safe distance from the boundaries.

A. Alternative Sensors that can be used in Navigation

- Color Sensor Module can be used to detect the provision markers when the robot moves over it.
- IR proximity sensors and Sharp IR range sensor can be used to detect obstacles and avoid them.
- Similarly, Ultrasonic Range Finder, Bump sensor and Reflective Optical sensor can be used to detect and avoid the obstacles in the arena.
- Linear motion sensor can be used to control linear motion in closed loop control and rotational motion sensors can be used to avoid rotational errors.

ACKNOWLEDGMENT

I thank the faculty and members of eRTS Labs, KReSIT, Indian Institute of Technology Bombay and eYantra Lab, BMS College of Engineering for providing the FireBird V ATmega 2560 robot and the accessories to carry out the project as a part of the e-Yantra Robotics Competition Plus 2014-15.

BIBLIOGRAPHY

1. Arun Mukundan, Introduction to OpenCV in Python, eYRC Plus 2014, IIT Bombay, 2014.
2. Amit Patel, Introduction to A*, Red Blog Games, 2014.
3. Laurent Luce, Solving Mazes using Python, Laurent Luce's Blog, 2011.

A video demonstration of this project can be watched online here: Project Video Link