Autonomous System Enabling Node and Edge Detection, Path Optimization, and Effective Color-coded Box Management in Diverse Robotic Environments

Deeksha Jayath[1], Chetna Mundra[2], Dr. Sowmya HK[3] and Dr. Sonia Maria Dsouza[4]

[1, 2, 4] BMS College of Engineering, 560019 Karnataka, India
deekshajayanth02@gmail.com, mundrachetna@gmail.com, s1985md@gmail.com
[3] New Horizon College of Engineering, 560103 Karnataka, India
sowmyahk2020@gmail.com

Abstract. This research paper introduces an innovative system designed for optimizing operations in industrial warehouses and similar environments. The system utilizes visual input, capturing the warehouse layout along with designated pick-up and drop-off zones tailored to specific industry requirements. Through image analysis, the system extracts critical data and employs a shortest path planning algorithm to determine the most efficient sequence of pick-up, drop-off, and navigation tasks that align with the given specifications within the designated setup. Once the optimal path is determined, the system generates a sequence of precise instructions to guide a robot through the required operations. These instructions are comprehensible to the robot, which executes them as it encounters nodes, junctions, dead ends, or obstacles in its path. This research presents a powerful tool for streamlining warehouse operations, enhancing productivity, and ensuring efficient task execution in complex industrial settings.

Keywords: node detection, edge detection, Path planning, colour-coded box handling, navigation commands, Lidar integration.

1 Introduction

This research introduces a system that uses visual input and algorithms to optimize operations, reduce expenses, and enhance productivity in a warehouse or a similar setup. This system comprehends layouts, extracts crucial data through image analysis, and employs path planning algorithms to streamline tasks.

This innovative approach aims to enhance warehouse efficiency in industrial settings. It involves three different types of images as inputs. The first type illustrates the basic layout with junction points (Fig. 1(a)). The second type incorporates colored boxes to represent various packages (Fig. 1(b)). The third type divides the warehouse into sections, like manufacturing (red), collection (green), distribution (blue), and drop-off (yellow), using colors (Fig. 1(c)).

2 Literature Review

Farag et al., [1] introduces a lane-lines detection and tracking system tailored for Advanced Driving Assistance Systems (ADAS) and self-driving cars, emphasizing simplicity and fast computation. While their algorithm successfully detects straight lane

lines, it faces challenges with light-colored roads and complex shade patterns, and it does not handle curved lanes effectively. In contrast, our proposed system offers a color-independent approach, providing consistent and precise lane detection, which also excels in curved track detection.

Wang et al., [2] presents an algorithm that achieves an impressive O(n log n) complexity for identifying a single route. Nonetheless, when our application demands the calculation of numerous routes and repeated determinations of the shortest distance, a different perspective emerges. In our paper, the algorithm's performance becomes contingent solely on the number of vertices, rendering it more efficient in addressing our specific requirements by computing the routes of the path. Consequently, our approach offers a more favorable solution for these extended use cases.

3 Methodology

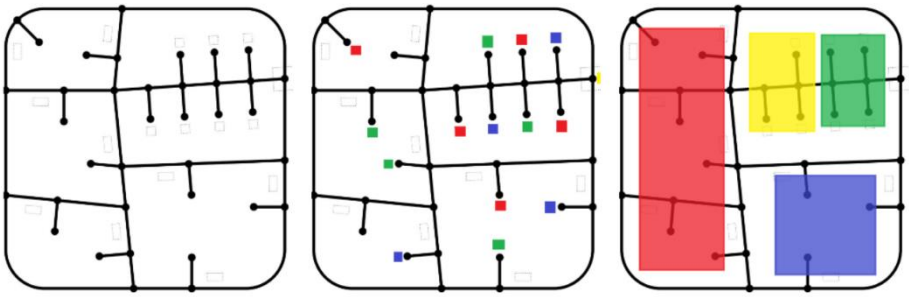

Fig. 1. (a) Basic Layout (b) Colour-coded Boxes (c) Colour Coded Areas

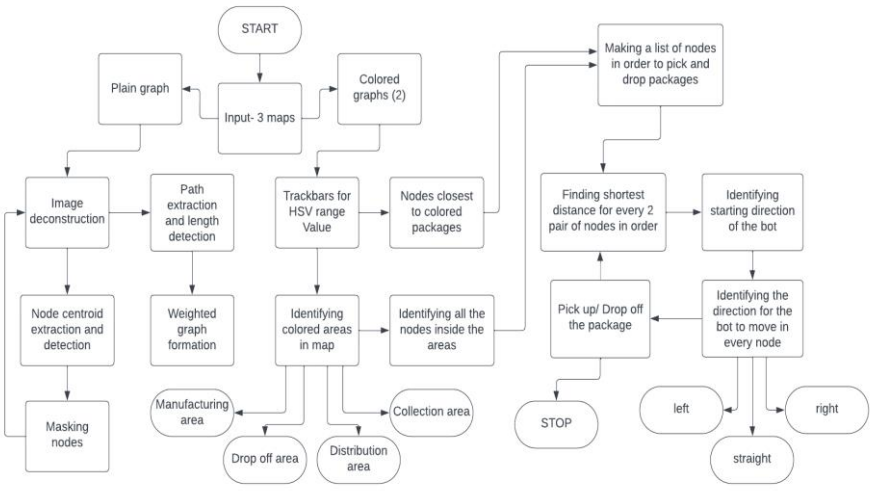

Fig. 2. Flowchart

3.1 Extracting Necessary Features and Information from the Given Image

The primary focus is on detecting features in these three scenarios. Firstly, in the basic map, it identifies nodes and tracks and accurately calculates their distances (Fig. 3(a) and Fig. 3(b)). These nodes are pivotal for navigation, and their coordinates guide subsequent actions. Subsequently, the original image's nodes are concealed, leaving behind only the black lines representing tracks (Fig. 3(c) and Fig. 3(d)). This simplifies the process of calculating precise track distances, which are later used as "weights" for the respective paths (Fig. 6). All this information is neatly organized in an index to pair nodes with the endpoints of specific tracks for efficient storage and retrieval.

Image processing:

- the given image is converted to greyscale using a weighted sum of the red, green, and blue channels. Grayscale pixel value –

$$(G)=0.299*R+0.587*G+0.114*B \quad (1)$$

- On the greyscale image low-pass filter that reduces noise and smooths the image is applied which uses the filter kernel G(x, y) to perform convolution with the grey-scale image, where σ is the standard deviation.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2-y^2}{2\sigma^2}} \quad (2)$$

- Further dilation is applied which is a morphological operation that is used to expand bright regions in a binary image to enhance or emphasize nodes initially and tracks in the later stages. Dilated image

$$(x,y) = \cup\,(i,j) \in B\; Image(x+i, y+j) \quad (3)$$

- Then we perform edge detection on the obtained image. This done by calculating the gradient magnitude of each pixel using convolution with Sobel filters first, then non-maximum suppression is applied to thin the edges and keep only the local maxima in the gradient direction. For each pixel, the algorithm checks if the gradient magnitude is a local maximum along the gradient direction. If it is, the pixel is preserved; otherwise, it is suppressed (set to zero). Finally, pixels with gradients above the high threshold are strong edges, while those below the low threshold are non-edges. Pixels between the two thresholds are weak edges, and they are strengthened by joining them to nearby strong edges, resulting in a binary image highlighting final detected edges. This finally gives the nodes as faint point as shown in Fig 3a, this is enhanced by employing the dilation operation.
  Gradient magnitude

$$(|G|) = \sqrt{Gx^2 + Gy^2} \quad (4)$$

  Here, Gx and Gy are the gradients in the x and y directions, which are computed using convolution with Sobel filters.

- From the obtained spots representing the node points we extract the centre points of the nodes to the coordinates the nodes on the given map by calculating moments using

$$\overline{x} = \frac{m01}{m00} = \frac{1}{A}\int_a^b xf(x)dx, \overline{y} = \frac{m10}{m00} = \frac{1}{A}\int_c^d yf(y)dy \quad (5)$$

- By overlapping the original map image and the obtained node contours the nodes are masked, then on this image the above-mentioned image processing is applied in the same order.
- After the dilation step length of the tracks is calculated by calculating the perimeter of the obtained contour after following all the image-processing steps. The perimeter is calculated by adding the Euclidean distances between consecutive contour points. Perimeter –

$$(P)=\sum_{i=0}^{N-1}(x_i - x_{i+1})^2+(y_i - y_{i+1})^2 \quad (6)$$

- The calculated coordinates of the centre points of the nodes and lengths of the tracks are stored in dictionary data structure with makes it easy to be accessed while applying path planning algorithm to find the shortest path for the assigned task which is explained in section 3.2.
- Additionally, the system extends its capabilities to process layouts featuring colored boxes and patches (Fig. 1(b) and Fig. 1(c)). A dedicated color detection function verifies if the color aligns with the bot's command requirements. Four colors are considered: red, yellow, blue, and green, each assigned distinct roles and packages.
- Trackbars are utilized to define H-Hue, S-Saturation, and V-Value (Brightness) to arrive at their suitable ranges for a given colour. These ranges play a role in identifying box colors and subsequently categorizing them. Furthermore, color coding is employed to designate the manufacturing, distribution, collection, and drop-off areas (as demonstrated in Fig. 1(b)).

The entire process of image processing and associated functions is autonomously applied as required. The program integrates image processing, node and lane (referred to as tracks or edges) detection, and the extraction of essential information from the three input images. These tasks are executed autonomously, eliminating the need for human intervention. Consequently, the program independently carries out tasks and generates commands for the robot's navigation, without any human interference.

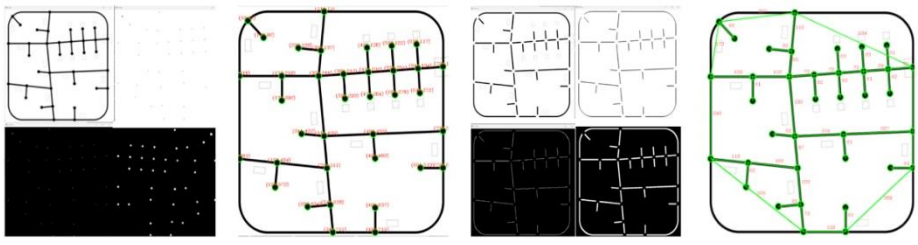

Fig. 3. (a) Node Detection (b) Coordinates of Nodes (c) Edge Detection (d) Weights of Edge

3.2 Path Planning and Navigation

Path Planning

A sample dictionary is produced with all of its zones, respective nodes, and associated colours, looking somewhat like this: {'manufacturing unit': {3: 'blue', 5: 'blue', 16: 'green', 30: 'red'}, 'drop off': {17: 'green', 18: 'red', 31: 'blue'}, 'collection area': {19: 'green', 20: 'blue', 32: 'blue', 33: 'red'}, 'distribution area': {2: 'blue', 8: 'red', 11: 'green'}}

The ideal design should have the fewest leaps possible between nodes. The order in which the packages should be picked up and dropped off is crucial to minimising the distance that the bot must travel. As the bot's starting node is already known, we must figure out how far it is from the nodes that make up the manufacturing unit or collection unit. The bot travels to the node that is closest to its starting point. Once the bot has reached its target, the second node changes from being the destination node to being the source, which is either the distribution area or the drop-off area. The matching package is subsequently picked up and delivered by the bot. The third node now serves as the source node in this procedure, and the subsequent package is chosen in the same way as before. A node gets eliminated from the dictionary each time it is encountered. The shortest distance to choose the next package is determined by iteratively updating the shortest path distances between all pairs of vertices by considering all possible intermediate vertices until the shortest paths are determined.. A list is maintained with all nodes that contain packages and need to be dropped off in order.

The shortest path between every two nodes can be calculated efficiently by traversing the neighbouring nodes level by level. If it encounters a node repeating, it dsicards the path and if the weight of the path is same as in the above algorthim, then it chooses that path. List data structure is used to store the nodes in the order they need to be visited.

Navigation

Once the list of nodes is prepared, the bot receives specific instructions whenever it encounters a node. These instructions dictate actions such as picking up a package, dropping off a package, starting, stopping, turning left, right, or proceeding straight. The mathematical computations involved in these instructions rely on calculating the sine of angles formed between two lines.

The initial orientation of the bot is determined based on its upcoming coordinates. Subsequently, the sine of the angle between three points, denoted as A1(x0,y0), A2(x_1,y_1), and A3(x_2,y_2), is computed. If the angle is approximately 180°, the sine value will be close to 0, indicating that the bot should continue straight. If the sine value is close to 1 or -1, it signifies that the bot should turn left or right, respectively from the table 1. When the sine value of the angle deviates from 0, 1, or -1, it implies that the path is curved.

Fig. 4(a) illustrates various scenarios involving points A, B, and C, which help determine the bot's directional choices. To decide whether the bot should turn left or right,

given that the minimum distance between any two lines is at least d, Table 1 outlines the conditions that govern the bot's direction.

Table 1. The row shows the direction of bot it is currently in, while the column shows the direction of bot in which it has to move forward to.

	x_1-x_2>d (left)	**x_2-x_1>d (right)**	**y_1-y_2>d (up)**	**y_2-y_1>d (down)**
x_0-x_1>d (left)	-	-	Sin(α) ≅1 right	Sin(α) ≅-1 left
x_1-x_0>d (right)	-	-	Sin(α) ≅1 left	Sin(α) ≅-1 right
y_0-y_1>d (up)	Sin(α) ≅ -1 left	Sin(α) ≅1 right	-	-
y_1-y_0>d (down)	Sin(α) ≅-1 right	Sin(α) ≅1 left	-	-

To identify curved edges, it is important to consider whether α represents an acute or obtuse angle. In cases involving curved edges, we examine whether the slope of the line is exactly 1. A slope of 1 indicates that it is the line originating from the curved edge, as demonstrated in Fig. 4(b). The directional instructions for this line can be derived from Table I. This approach is efficient because the tangent to the curve also possesses a slope of 1. Consequently, line 34-35 (Fig. 4(b)) is perfectly perpendicular to the curve. In Fig. 4(b), the angles between the lines represented by (4,1,10), (8,0,2), and (36, 28, 27) are all acute angles when compared to each other. If the slope of the second line is not equal to 1, it means the bot should proceed straight, and the initial position of the bot must be recalibrated. Once the initial direction of the bot is adjusted, we can consult Table 1 to determine the navigation instructions.

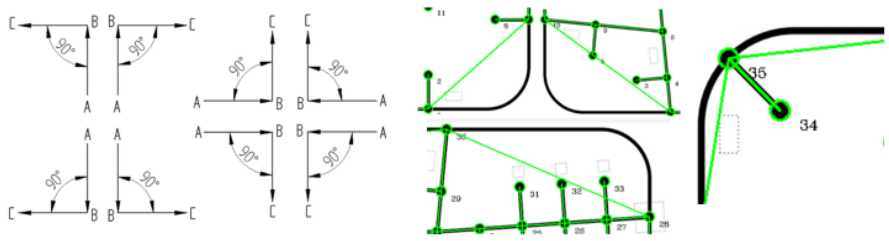

Fig. 4. (a) Possible Direction of movement and (b) Special Curved Regions

4 Implementation using LiDAR-

Integrating LiDAR technology into the line-following robot enhances safety and functionality. LiDAR sensors continuously scan, creating real-time 3D maps for obstacle detection and comprehensive setup views. This proactive obstacle response minimizes collision risks and equipment or goods damage. LiDAR's precision in mapping

and localization ensures accurate path adherence and destination reach. It enables dynamic path planning for adaptability to changing setup configurations. LiDAR also aids object recognition and classification, benefiting tasks like optimizing picking, packing, and inventory management. It operates effectively in varying lighting conditions, guaranteeing consistent performance.

5 Comparitive Study

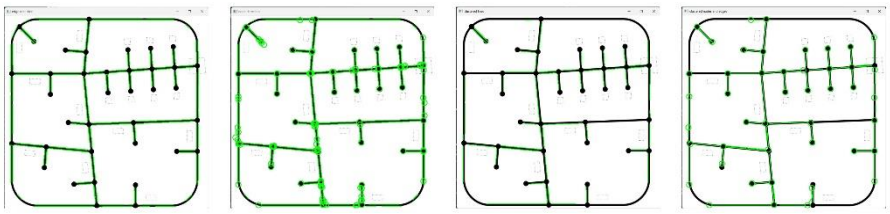

Fig. 5. Detecting nodes and edges using hough technology – (a) Edge Detection (b) Node Detection (c) Clustered Edges (d) Clustered Edges and Nodes

In this approach, we utilize existing Hough technology [1] but encounter issues such as redundant line detection and the inability to identify curved lines. We mark nodes at line ends and cluster lines and nodes to obtain approximate coordinates. However, this yields redundant nodes and fails to identify curved edges. Our algorithm excels by efficiently detecting edges, nodes, and edge weights, shown in Fig. 3(b) and Fig. 3(d). The difference between nodes and edges identified in Fig. 5(d). and Fig. 3(d). is evident.

6 Results

The procedure takes images of maps as its input and generates a sequence of directives, denoted by 'S' for Straight, 'R' for Right, 'L' for Left, 'U' for U-turn, 'P' for Pick-up, 'D' for Drop, and 'O' for Stop. These directives are transmitted to the robot via Bluetooth, allowing it to traverse nodes and carry out the appropriate actions to reach its intended destination. The ultimate goal of this problem can be articulated as follows:

```
['S', 'S', 'S', 'S', 'S', 'R', 'L', 'P', 'R', 'L', 'S',
'R', 'D', 'R', 'S', 'L', 'P', 'L', 'R', 'S', 'R', 'D',
'L', 'S', 'L', 'S', 'S', 'S', 'S', 'S', 'L', 'P', 'R',
'S', 'R', 'D', 'R', 'S', 'R', 'P', 'L', 'S', 'S', 'L',
'L', 'R', 'D', 'L', 'L', 'S', 'R', 'P', 'L', 'S', 'S',
'R', 'S', 'L', 'D', 'R', 'S', 'L', 'S', 'R', 'L', 'P',
'R', 'L', 'S', 'R', 'S', 'L', 'D', 'L', 'S', 'R', 'P',
'R', 'R', 'S', 'S', 'R', 'D', 'R', 'R', 'S', 'S', 'S',
'R', 'S', 'S', 'L', 'P', 'R', 'S', 'S', 'L', 'S', 'S',
'S', 'L', 'L', 'D', 'L', 'S', 'S', 'O']
```

Fig. 6. Sequence of Instructions

7 Future Scope and Conclusion

The suggested autonomous system offers notable advantages in diverse operational scenarios by enabling precise node identification, centroid computation, edge recognition, distance measurement, and shortest path planning. It generates autonomous commands and leverages color-coded regions for enhanced box handling. Using computer vision, it improves navigation efficiency, reduces human errors, and prioritizes tasks, thus enhancing operational efficiency in manufacturing, distribution, and collection areas. This comprehensive solution has the potential to significantly boost efficiency in various settings. Looking ahead, this system promises for expansion and innovation beyond warehouses. Its adaptability allows it to serve diverse environments like supermarkets, factories, logistics centers, and even outdoor spaces. It can incorporate advanced technologies such as computer vision, RFID tracking, ultrasonic sensors, or AI-driven algorithms, ensuring it remains at the forefront of automation and navigation solutions for various sectors.

References

1. W. Farag and Z. Saleh, "Road Lane-Lines Detection in Real-Time for Advanced Driving Assistance Systems," 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), Sakhier, Bahrain, 2018, pp. 1-8, doi: 10.1109/3ICT.2018.8855797.
2. Wang, H., Xue, J. Near-Optimal Algorithms for Shortest Paths in Weighted Unit-Disk Graphs. Discrete Comput Geom 64, 1141–1166 (2020). https://doi.org/10.1007/s00454-020-00219-7
3. Jingkuan Song, Zhilong Zhou, Lianli Gao, Xing Xu, and Heng Tao Shen. 2018. Cumulative Nets for Edge Detection. In Proceedings of the 26th ACM international conference on Multimedia (MM '18). Association for Computing Machinery, New York, NY, USA, 1847–1855.
4. Y. Wan, S. Jia and D. Wang, "Edge Detection Algorithm Based on Grey System Theory Combined with Directed Graph," 2013 Seventh International Conference on Image and Graphics, Qingdao, China, 2013, pp. 180-185, doi: 10.1109/ICIG.2013.42.
5. Faris Adnan Padhilah and Wahidin Wahab. 2018. Comparison Between Image Processing Methods for Detecting Object Like Circle. In Proceedings of the 2018 International Conference on Digital Medicine and Image Processing (DMIP '18). Association for Computing Machinery, New York, NY, USA, 33–36.
6. Sabine Storandt. 2018. Sensible edge weight rounding for realistic path planning. In Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18). Association for Computing Machinery, New York, NY, USA, 89–98.
7. M. Aasim Qureshi, Mohd Fadzil Hassan, Sohail Safdar, Rehan Akbar, and Rabia Sammi. 2009. An edge-wise linear shortest path algorithm for non negative weighted undirected graphs. In Proceedings of the 7th International Conference on Frontiers of Information Technology (FIT '09). Association for Computing Machinery, New York, NY, USA, Article 67, 1–4.