Autonomous Navigation Vehicle Using Natural Language

CH.koteshwar
21241A0476
department of Ece
Griet
Hyderabad,India

M.Durga Chaithanya
21241A04A1
department of ECE
Griet
Hyderabad,India

MD.Abdul kaleem
department of ECE
21241A04A2
Griet
Hyderabad,India

DR Arvind Vishnubhatla
department of ECE
professor
Griet
Hyderabad,India

***Abstract*—Our project aims to develop a voice-controlled robotic vehicle that can navigate various locations according to user's Natural language input (can be text/speech). Our system aims to accurately understand and traverse most complex workflows. Our system is capable of communicating with a user according to the given task. We considered LLMs (Large language models) as a brain, made use of their capabilities to make the system understand human inputs, design workflow flow and communicate with the user. We considered every possible Case like (task in task , task after task) to make it more efficient. We used A* algorithm for pathfinding and Made use of Arduino with a motor controller for path following.**

Transport, travel, and communicationion are some major features where this robotic vehicle shines. The main purpose is to Help humans in trasposrting items and transferring information from one place to another with natural language input. It not only travels by commands, but the user can also select locations where they like to have the vehicle. We inspired from Robotic vaccume cleaners where there been lot of advancements into home localisation and obstacle avoidance. We aim to integrate Voice control and communication feature to enhance its abilities

I. Introduction

For the past decade, we have been seeing a lot of advancements in the home localisation fields especially towards vacuum cleaning Robots. The obstacle detection, path following, and mapping had been pretty accurate. Robots can efficiently navigate from one location to another. Ecovacs DEEBOT X1 OMNI is one of the Vacuum Cleaning Robots that have crossed a lot of benchmarks towards obstacle avoidance, mapping and path-following. One interesting thing is the integration of voice assistants. Which can process natural language and convert it into commands.

We aim to build a robot that not only cleans but also travels, transfers and speaks according to human commands. We built a system that efficiently converts the human natural language into a set of tasks and locations to travel. The robot aims to travel to the given location and speak out according to the given task. We tried to utilise the capability of LLMs(large language models) to accurately convert given human messages into tasks and locations. We designed the LLM's architecture in such a way that it can process and transfor complex workflows. We implemented a decoder algorithm to transform the outputss from the LLM architecture to signals that the vehicle can travel.

We utilised the A* algorithm to find the shortest path between nods that a user wants the vehicle to travel. As the vehicle just can transport, travel and speak, it can't do other tasks by itself. We made the robot speak so that it can ask for other tasks to be performed by the user. The decoder algorithm is capable of generating signals and angles so that Arduino with a motor controller and a gyroscope can travel accordingly.

All The functionalities are containerised using docker so that the user only has to give input in the form of text or speech, and the container outputs directions and angles to follow. The directions and angles are sent through the serial port to Arduino. The docker container is made to run on Raspberry Pi with an Arduino connection so that everything can be processed wirelessly.

We aimed to make advancements in the field of human input processing and speaking capabilities of the robot. Utilising the obstacle avoidance features described in TensorFlow blog [https://blog.tensorflow.org/2020/01/ecovacs-robotics-ai-robotic-vacuum.html]. The remarkable progress in LLMs and obstacle avoidance helped to build this project.

II. PREVIOUS WORKS

In contrast to the conventional random navigation, laser-based mapping was developed in 2010 by the Neato Robotics XV-11 robotic vacuum, enabling navigation in straight lines.[1]. Both iRobot and Dyson debuted camera-based mapping in 2015.[2][3].As of 2018, impediments like cables, shoes, and dog poop are still highly challenging for robots to avoid.[4][5]. The first natural language home robot using Al speech interaction and control technologies was the DEEBOT-X1 Family with YIKO[20] speech Assistant, introduced by ECOVACS in 2022.[6][7][8]. SwitchBot claimed to have the smallest robot vacuum in the world when they unveiled the K10 Plus in 2023 [9].[10][11]

Marking significant advancements in autonomous technology. The problem of the shortest path in graph theory involves finding a path between two nodes such that the total weight of its edges is minimized. Several algorithms address this

efficiently: Dijkstra's algorithm is optimal for graphs with nonnegative weights (Dijkstra, 1959) [12], while the Bellman-Ford algorithm handles graphs with possible negative weights (Bellman, 1958)[13]. The A* search algorithm improves efficiency using heuristics (Hart, Nilsson & Raphael, 1968)[14]. For all-pairs shortest paths, Our approach focuses on developing a fast and efficient autonomous navigation system suitable for limited space environments such as homes, industries, workplaces, and educational institutions. Existing autonomous vehicle solutions are costly and unsuitable for compact spaces.

III. VEHICLE FUNCTIONALITIES

A. *3.1 Understanding Natural language*

This is a crucial step in the generation of workflow plans and responses. We use LLMs with tool calling features to accurately understand user response and make plans accordingly. We make use of LLM in different ways
Tool-calling LLM (for Generation of task-specific workflows)
Input Classification LLM (for classifying whether the user is asking to perform a Task or the user question refers to a general response like the current position of the vehicle or history of tasks it has performed till now or any functionality related)
Orchestrator (For making user Input a bit more constructed to make perfect plans)
Responding LLM (for responding to the input message given by user. It doesn't refer to location-specific messages)

B. *3.2 indoor positioning*

We will be focusing on using the A* Algorithm. A* is the most popular choice for pathfinding because it is fairly flexible and can be used in a wide range of contexts. A* is similar to Dijkstra's Algorithm in that it can be used to find the shortest path. A* is also similar to Greedy Best-First Search in that it uses a heuristic to guide itself. In the simplest case, In this instance, it is just as quick as Greedy Best-First Search.

We make the A* algorithm to find the shortest path by considering eight directions (Left, Right, Top, Bottom, Top-Left, Top-Right, Bottom-Left, Bottom-Right). The path returned by A* will be in two formats: coordinates and directions.

Coordinates: Used to locate the traced path on the map.
Format of Coordinates: (1,2) $\rightarrow$ (1,3) $\rightarrow$ (2,4)

Directions: Used to send movement signals to the Arduino.
Format of Directions: (Left, Left, Left-Down)

To clarify the format of directions, the A* algorithm considers the user's graph perspective. When moving forward in the leftward direction, it continuously outputs Left. If a turn is required to the right from that position, the output will be Down, as the movement occurs downward in the graph representation.

Correct indoor positioning is a crucial aspect of autonomous navigation. While techniques such as GPS, anchor-based tracking, and RFID tags are commonly used, they tend to be costly and require significant maintenance. Given that our system operates within a confined environment with a predefined layout, a more efficient and cost-effective approach is needed

C. *3.3 decoder*

The decoder is specifically used to make the outputs generated by the A* algorithm in a way that can be instructed to Arduino so that the vehicle follows that specific path accurately without any directionality errors.**we had abrivated about decoder in the human input processing in section 5.2**.

D. *3.4 Transferring things from one place to another*

Here we use a combination of the A* algorithm and decoder.
A* algorithm - It is a path-finding algorithm. it is highly accurate in finding the shortest path between 2 nodes in graph-based environments. In our case, we consider our image as a graph where obstacles are represented by 1s and free paths are represented by 0. We make use of the A* algorithm to find the shortest path between the nodes given by the user(through LLMs).
we can see the more information about A*algorithm in section pathfinding algorithms 5.1

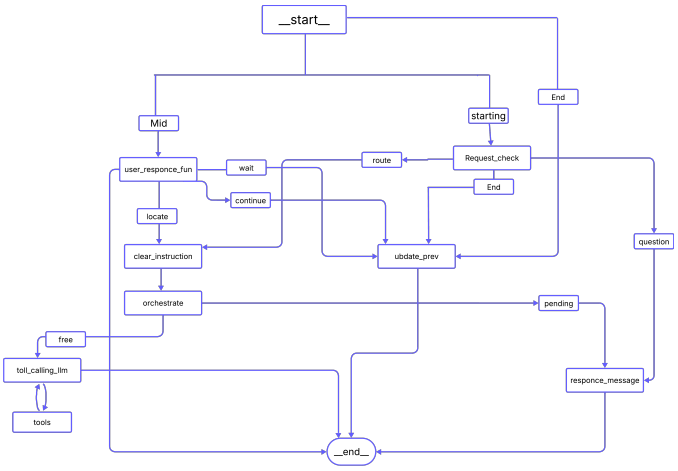
E. *3.5 Responding to user Inputs*

There are 3 major reasons for implementing this functionality.

1.What if the vehicle performs another task while the user requests his task?
2.What if there is no such location in the house or industry environment?
3.What if the user wants to clarify things like (where it is currently or the history of tasks it has performed till now or any functionality related) rather than making it perform the task?
Section 4.2

IV. HUMAN INPUT PROCESSING

Let's now look at how the model responds when a user inputs. We utilised the capability of Large Language Models (LLMs) to respond to the user input by incorporating tool calling, Orchestrating and reponce mapping features of LLM's. In a nutshell, we are making the LLM communicate according to the task, make travelling plans according to the user request and make deccisions about input mapping.



A. *4.1 Travel plans*

We make use of 2 agents
1. orchestrater
2. tool calling LLM

Orchestrater
So the function of the orchestrater is to make user input more understandable. Orchestrater is responsible for formatting user inputs and corresponding tasks into a JSON format. This helps Toolcalling LLM to better understand and make plans. So JSON format includes number of tasks named task 1, task 2 etc.each task is a supperate dictnery wheer route and its task specific duty of the vehicle.

Toolcalling LLM
To speak and decide routes for given locations. Tools are integrated into the LLM. LLM decides which tool to call according to the given prompt. The primary functionality of the vehicle is to travel and speak. Such that it can't do other tasks by itself it needs to speak out with humans to perform critical tasks. It uses a speak tool to communicate with humans if needed and a Travel tool to go from one place to other. LLM decide when to travel and when to speak.

B. *4.2 Input Mapping*

The vehicle is designed to travel and perform various tasks. However, it is important to consider not only the tasks it needs to perform but also the timing of these tasks. The key point is that some tasks can be performed sequentially, while some can be done in parallel, some requires only communication and some require only travelling. The question then becomes: how do we determine the appropriate approach? This is where input mapping comes into play.

For input mapping, we have to implemented concept of states. so states decides corresponding actions needs to be taken place.
sent: bool (represents whether the previous instruction is executed or not)
previous_state: str (if the vehicle travelled the executed route/ not)
currrent_mode. : str (whether the input is to free up the current state or to perform now task)
question_type: str (user input is about travelling or to answer any task-related questions.)

Let's first classify the inputs as cases and determine how each case is tackled using these states.

Case 1: if the vehicle is not performing any tasks and the user needs to get some work done.
In this case, we consider the previous state as free and sent as true. that means the input can be mapped to the orchestrater as the task can be executed successfully we set the stent as true. then, until the vehicle travels the entire path, we set the previous state as pending.

Let's say the given task is done successfully without any obstacles then we free up the previous_state explicitly.
Case 2: If a new independent instruction is given while a previous instruction is still pending
the vehicle prefers performing the instructions sequentially. Since the previous instruction is awaiting execution (Previous_state="pending") but has already been sent for processing (sent= True), we handle the orchestration accordingly. However, we cannot proceed with toll calling, as this would send travel information to the vehicle that is already committed to another task. To explain this situation, we utilize a responsive language model to explain the scenario.

Once the previous task is successfully traveled then we try to free up the previous state. While freeing up if sent is not true (as mentioned in case 2) that means there is a task waiting to be executed so it makes sure that task is sent to the vehicle to travel and sets previous state = pending and sent = true. After the successful execution of 2nd task then we free up previous_state. In this way, independent tasks will be performed sequentially.

Case 3: If the task is instructed which is part of the previous parent task
In this case, we have to perform the present task and then continue performing the current task. To make this happen we Introduced question_type so it decides whether the user input asks to travel or to proceed with the current task. But note this decision only happens when the vehicle is asking questions (imagine the vehicle asks to make a coffee but you instruct the vehicle to take the powder from the store room and come back) in this case the vehicle is asking and in a stopped state in the middle of the parent task and you are instructed another task. We used recursion to implement this scenario. where it is sent back to the model and the vehicle travels the path. After completion, the vehicle continues to perform the parent task.

if the question type is location in the middle of the parent task then the vehcle travels the given locations and perform

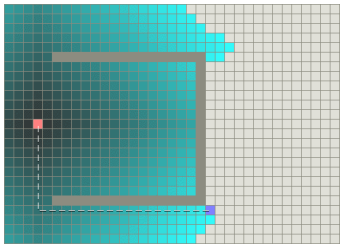
given tasks then continue to do the parent task. simple priciple is that if the instruction is given while travelling it is conciderd as the next task and if the instruction is given oin the middle when asked question by the vehcle it is performed as part of current task.

Case 4: what if the question not at all about travelling in this case using question classifiction by LLM it is direction sent to response LLM. response LLM has every information like location history, task hystory functionalites and locations it can travell. so according to the given input the responce making LLM gives responses.

V. PATH FINDING ALGORITHMS

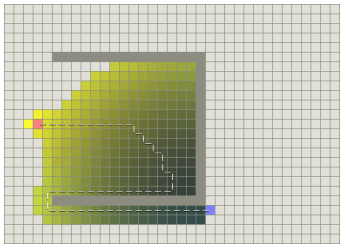
A. *5.1 overview of Existing Approches*

There are various types algorithms are avaliable to find the shortest path most of the people think of about dijkstra algorithm.dijkstra algorithm search from all the side and tell us which path is shortest.



dijkstra

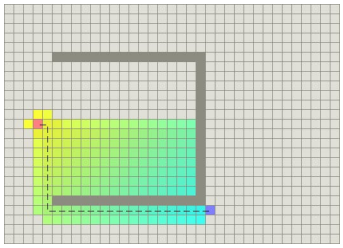
Bellman-Ford algorithm is also used for searching shortest path but it is slightly slower than the dikstra algorithm.If there are any negative weight in the graph or in tree it can handle it and find the shortest path.



best-first-search

The First-In-First-Out queue is used by the Breadth-First search algorithm. One node at a time is visited by this algorithm. Using the number of traveled edges as a proxy for distance, this Breadth-First search algorithm explores nodes in order of their separation from the source node [15] [16].
Heuristic search mimics the cognitive model of the human brain, which weighs some options before reaching a decision rather than all of them. When you need the shortest pathways between every pair, you utilize the Floyd-Warshall and Johnson algorithm [15][16].

1) : A*algorithm



A*

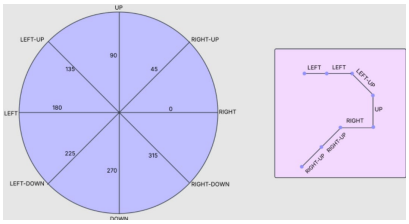
A* algorithm is used for finding the shortest path between the source node to destination node.The algorithm is tested by using images that represent a map which belongs to images that represents in a binary format. The map is converted into black and white.Black is a obstacle and white is for the free space[18].

the amount of time taken to travel from one location to another can be known by finding the number of nodes the Algorithm is visited. as we are aiming to find the path that takes least amount of time higher shortest path accuracy. the A* algorithm utilities two lists the open and closed list. where the open list has all the nodes that are not yet visited and closed list contains the nodes that has been visited already. so, marker array is made to determine whether a state is presnt in open list or closed list[19][20].

To discover the shortest path more efficiently, the A* method is utilized. And it also calculate the total distance cost and node-to-node cost it is so extraodinary feature in A*algorithm.A*Algorithm main components Are g(n) and h(n),in g(n) it calculate the cost from Start node to current node. And in the h(n)[heuristic] it estimate the cost from current code to final node or goal node.By joining the g(n) and h(n) we can estimate the cost from intial node to goal node .

f(n)=g(n)+h(n).

B. *5.2 Decoder*



Till now we have focused on how natural language is transformed into locations and finding the shortest path between locations (nodes). the challenge is how we can make the vehicle follow the path. one obvious way is to send the signals to Arduino with motors and motor drivers. as the Arduino contains the code to make sure the motors rotate as per the signals.

but what kind of signals can we send? the signals must be in such a way that they align with the code in the Arduino (this is described in the Arduino section). alignment here means that the directions must be accurately followed by the vehicle. Arduino code just contains the functionalities like (L, R, F, S) where

1. L - Left
2. R - Right
3. F - Forward
4. S - Stop

But outputs of the A* algorithm are like (LEFT, RIGHT, RIGHT-UP,..) and it just returns to us the direction it is following. As you can see in the FIG (...) for a sample path the A* algorithm output doesn't describe any turns. as RIGHT-UP to RIGHT, the Vehcle need to rotate about clockwise 45 degrees and RIGHT to UP is about Anti clockwise 90 degrees. As the vehicle rotates in the direction (CLOCK-WISE or ANTI-CLOCKWISE) where the angle of rotation is less.To make sure that we are considering both Anti-Clockwise and Clockwise. Anti-clockwise indicate turning left and clockwise indicates turning right. But it is only applicable when there is a change in direction like Right-UP to Right, Right to Up etc.. if there is same direction such as RIT-UP to RIGHT-UP, LEFT to LEFT we represent it as Forward (f). Clockwise 45 can be known to be one time right as one right and one left turn represent 45 degrees clockwise and anticlockwise. the same way Clockwise 90 is two times right, Anti clockwise 90 is two times left as signals to the Arduino.

if we apply the same algorithm to the above example by considering it initially in the right direction

we basically follow two steps to make this happen

Direction to angles
Ange to signals.

A* algorithm output	RIGHT	RIGHT-UP	RIGHT-UP	RIGHT	UP	LEFT-UP	LEFT	LEFT
Angle Decoding	A-C 45	f	C 45	A 90	A 45	A 45	f	f
Signal Decoding	L F	F	R F	L L F	L F	L F	F	F

VI. IMPLEMENTATION

the user interface is built by HTML,CSS,JS . First page will contain an image where user needs to select and click on proceed. Ths image is nothing but a home architecture. where we need to perform our tasks. on click proceed the layout truns in to a binary format like 1's and 0's .the 1's are assumed as obstacles and 0's are assumed as free space the vehicle travels in the free space by the a*algorithm it find the shortest path.

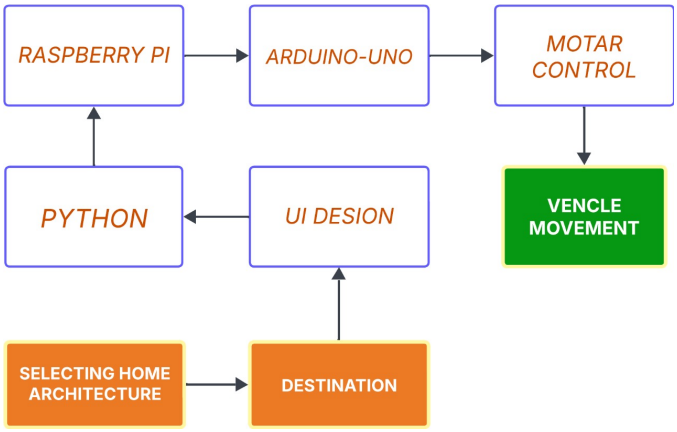


Fig. 1. System Architecture

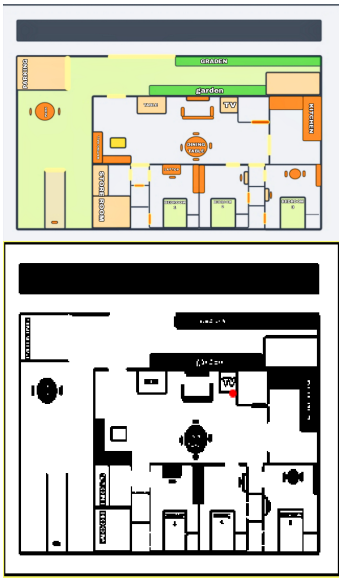


fig:virtual layout to binary

The user can give voice command or text input or can locate the starting node and final node. then modes decides where to travel and what to speak if the input is text or speech then by the A*algorithm it finds the shortest path.The information given by A* algorithm is sent into decoder such that it converts into angles and signals to be followed by vehicle.

we are containerizing the entire functionality using docker. By employing this make giving input and getting output will be more easier especially while integrating to JS (java script). everything is serially processed using async and await features provided by js.

As commends are serially extracted by JS it is parallel sent to Arduino using serial port connection. then code in the Arduino controls mortars to efficiently follow the given paths. we try to implement on raspberry pi so wire less environment will be created.
A critical aspect of this implementation is seamless

communication between the Raspberry Pi and Arduino. The A* algorithm outputs a sequence of directional movements (e.g., Left, Right, Top, Bottom), which must be translated into Arduino-compatible motor control commands (L - Left, R - Right, F - Forward, S - Stop). This translation is handled through Python, which encodes and transmits the instructions via a serial connection (PySerial). The Arduino then executes these commands in real-time, ensuring precise movement along the computed path. After encoding the A* algorithms output. Through pyserial connection in python we send this information step by step to Arduino. Arduino process those step by step. Also we use raspberry Pi to impliment the python code. We can operate the raspberry Pi os using our laptop. This way the path will be efficiently tracked to the destination

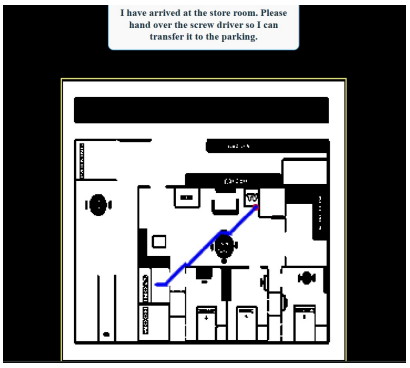
The central processing unit is the Raspberry Pi., running the path-planning algorithm and handling user input, while the Arduino controls the robot's movement. The Raspberry Pi can be accessed remotely via SSH or VNC, allowing easy monitoring and debugging.

VII. RESULT

vehicle trvels in only shortest path to reach the goal by using A*algorithm. .Vehicle trvels according to our instruction .we need to explain the instructio in vioice command or in a text command like

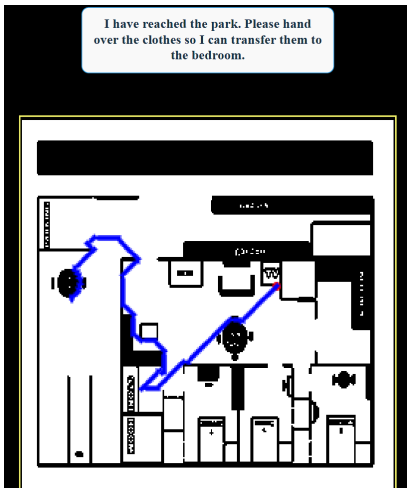
Example:

Go to storeroom and get the screw driver. Give it in a parking, then go to park, get the clothes and put it in the bedroom.

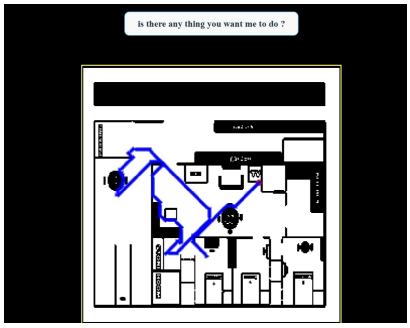


robot: I have arrived at the storeroom. Could any one hand me the screwdriver?
Human: I gave you the screwdriver.

Robot : I have arrived at the parking Please take the screwdriver
Human : done.



Robot: I have arrived at the park. Could you please give me the clothes?.
Human: I gave you the clothes



Robot : I have arrived at bedroom 3. Please take the clothes
Human : Thank you
In the middle of vehicle moving if we give any new instruction it will first complete what we gave at first later it go to next instruction

CONCLUSION

The vehicle is perfectly navigating the generated paths and accuracy of decision making and work flow designing with complex human inputs is also high. The Sorest path generation with least time is made possible using A* algorithm. The obstacle avoidance in highly objective environments can be improved, and this can be implemented by using LIDAR and some of the computer vision techniques. We are aiming to improve the obstacle detection through using light weight computer vision models and making multi-model LLM to process the input and make decision exactly how a human eyes sends visual input to the brain to make decisions.

we conclude our model can now make decision and responds to any human inputs and also can travel, transfer and communicate according to the given task. then algorithm can produce shortest paths in high obstacle environments

REFERENCES

1.”Deluxe brand Dyson creates its first robot vacuum, the 360 Eye”. 2014-09-04. Retrieved 2015-09-26.
2.IEEE Spectrum: Technology, Engineering, and Science News. 16 Sep 2015. Retrieved 2020-05-17.
3. ”Neato XV-11 robotic vacuum review”. Engadget. August 24, 2010. Retrieved 2020-05-11
4. Solon, Olivia (2016-08-15). ”Roomba creator responds to reports of the ’poopocalypse’: We see this a lot”. The Guardian. ISSN 0261-3077. Retrieved on 2020-05-17.
5.”iRobot Brings Visual Mapping and Navigation to the Roomba 980”. IEEE Spectrum: Technology, Engineering, and Science News. 16 Sep 2015. Retrieved 2020-05-17.
6.”iRobot says 20 percent of the world’s vacuums are now robots”. TechCrunch. November 2016. Archived from the original on 2020-12-15. Retrieved 2020-05-17.
7.”DEEBOT X1 Family — New Arrival All in One Vacuum and Mop”. ECOVACS US. Retrieved 2022-11-25. 8.”ECOVACS Launches The DEEBOT X1 Range Of Robot Vacuums & Mops In Australia”. 21 April 2022.
9.ECOVACS. ”WELCOME TO THE HANDS-FREE FUTURE: ECOVACS’ NEW FLAGSHIP PRODUCT, THE DEEBOT X1 OMNI, IS NOW AVAILABLE FOR PURCHASE”. www.prnewswire.com (Press release). Retrieved 2022-11-18.
10.”DEEBOT X1 Family — New Arrival All in One Vacuum and Mop”. ECOVACS US. Retrieved 2022-11-18. 11. ”SwitchBot Mini Robot Vacuum K10+ — 50% Smaller in Size”. SwitchBot US. Retrieved 2024-08-22.
12.Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik.
13.Bellman, R. (1958). On a routing problem. Quarterly of Applied Mathematics.
14.Floyd, R. W. (1962). Algorithm 97: Shortest path. Communications of the ACM.
15.Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics.
16.Johnson, D. B. (1977). Efficient algorithms for shortest paths in sparse networks. Journal of the ACM.
17.Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory.
18.Kumar N, Kaur S. Bidirectional Graph Search Techniques for Finding Shortest Path in Image Based Maze Problem. International Research Journal of Engineering and Technology. 2019
19. Jing, X., & Yang, X. (2018). Application and Improvement of Heuristic Function in A* Algorithm.
20.Kumar N. Bidirectional Graph Search Techniques for Finding Shortest Path in Image Based Maze Problem (Master’s thesis, Punjab Technical University). 2019;1411-1414.