International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 02 | Feb 2020 www.irjet.net p-ISSN: 2395-0072

ANALYSIS OF A-STAR BOT

Terrell Pereira[1], Edelquinn Mendes[2], Sabiya Shaikh[3]

[1,2,3]*Student, Department of Computer Engineering, Fr. C. Rodrigues Institute of Technology, Maharashtra, India*

Abstract - *The existing robots use sensors to traverse the maze in doing so there are certain problems that are encountered. These robots use a long process of training and cannot adjust to dynamic environments. The path found each time the robot solves the maze is different and hence it is not the shortest and the most efficient path. Our approach is to traverse a maze with obstacles using the shortest path using the A* Search Algorithm. The main aim is to make an Arduino based efficient autonomous maze solver robot which has the ability to navigate automatically in an unknown area without the help of any sensors. The advantage of this over existing systems is that the method proposed here involves image processing and path finding algorithms; which works faster because of acquiring the maze's data beforehand, rather than going through the maze cell by cell. The entire maze is captured to determine the possible paths and A* Search Algorithm is used to find the best route. This approach helps the robot observe the maze beforehand and intelligently evade traps and loops. Also, the cost and complexity of making the robot is less due to no use of sensors. Applications of such autonomous vehicles range from simple tasks like robots employed in industries to carry goods through factories, office buildings and other work spaces to dangerous or difficult to reach areas like bomb sniffing, finding humans in a wreckage [10], etc.*

Key Words: A* Search Algorithm, Image Processing, Maze Solving Robot, Path Finding,

1. INTRODUCTION

Existing maze solving robots have various sensors attached to them in order to traverse through a maze. But in places where these bots can be used these sensors can be easily damaged or rendered useless like in run down areas etc. In order to solve this problem, we need to build a bot that can traverse through a maze without the help of any sensors. It should also be able to calculate the shortest path and be able to move as quickly as possible. The aim is to build an autonomous robot that is capable of finding the shortest path in a maze and traversing through it without using sensors as a proof of concept that the maze solving problem can be solved without the need of real time inputs using the sensors but just by using an image of the maze.

The shortest path will be calculated using the A* Search Algorithm on the processed image. The final robot will be able to traverse the maze only with the help of an image taken of the maze. Using the image taken, the shortest path to get from one end of the maze to another will be calculated. The solution to the maze will be sent to the robot to let it traverse through the maze. In order to do this we will be using arduino IDE to create a link between the software and hardware. We will use various image processing techniques to convert the image into an image that can be taken as an input to find the shortest path using A* Algorithm. The communication between the bot and the software will be done using the bluetooth module in the form of a chain code.

2. LITERATURE SURVEY

Maze solving is an important field of robotics which is at least 30 years old. It tests the Decision Making Algorithm where the robot will be placed in an unknown environment and it will have to intelligently find its way. The micro-mouse has developed significantly over the years starting from the basic wall following algorithm [4] to using algorithms like Flood Fill [2][3].

There are various autonomous robotic devices that work on autonomous navigation have been employed to make the lives of human easier. The Image processing and Path finding using A* Search algorithm [5] provides an insight on how this can be done. In maze solving, the robot is placed in a confined area and it needs to explore its surrounding and find a way out accordingly. Although a maze may not be like anything we find in the real world it still is useful to study specific aspects of a problem. This is very similar to the robots in the Micro-Mouse [1] competitions that start from the corner of the maze and should reach the centre of the maze in the least time possible. These robots know the starting location and the target location but do not know about the obstacles in between. The robot needs to navigate through the maze avoiding the obstacles and traversing through the best possible path.

3. PROPOSED SYSTEM FOR MAZE SOLVING ROBOT

For creating a bot that can travel through a maze intelligently without human intervention the system will require the following modules to work efficiently. The Image Processing Module, Shortest Path Finder and the Bot itself. The role of each module is as follows:

- **Image Processing Module**
  The photo of the maze that is taken cannot be used directly to find the shortest path in the maze. This is because the image taken by the camera is not binary and may contain noise which will cause

errors in processing. Hence, the image needs to be processed before applying the A* Algorithm.

- **Shortest Path Finder**
  The A* Search Algorithm is used to find the shortest path to traverse in the maze while avoiding various obstacles. The most reliable and efficient path will be calculated and the data regarding the same will be sent to the bot.

- **The Bot**
  The Bot is constructed using an Arduino Uno. It has omni-directional wheels that will allow it to move smoothly in 8 directions fast and efficiently. It is connected to a Bluetooth module to send and receive signals and to interact with the software.

Combining all modules together we obtain an autonomous bot which is fast and efficient and can solve a maze in the minimum time.

3.1 Image Processing

Image processing [6][12] is an integral part of the system as the image is the sole input that is given to the system. A photo of the maze is taken by either a mobile phone or a webcam. This RGB image is then processed until we obtain a binary image where the obstacles and path are easily distinguishable.

The steps used in image processing are given below:

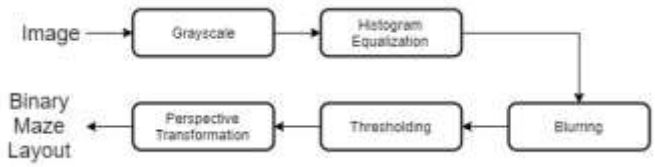

Fig -1: Image Processing Steps

- **Grayscale**

Fig -2: Grayscale Transformation

Digital images have three basic pixel color values viz red, green and blue (RGB). These values are called channels in digital images.

These channels each contain a brightness level that determine the shades of the colors. In grayscale, the colors are removed leaving only the intensity values of the image. Hence, the image appears to be monochrome.

- **Histogram Equalization**

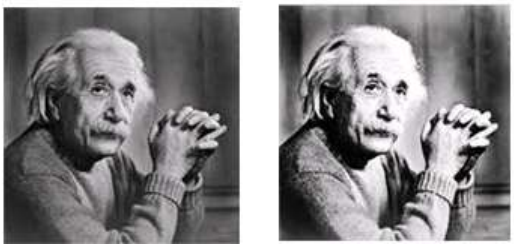

Fig -3: Histogram Equalization

The A* Search Algorithm is used to find the shortest path to traverse in the maze while avoiding various obstacles. The most reliable and efficient path will be calculated and the data regarding the same will be sent to the bot.

- **Blurring**

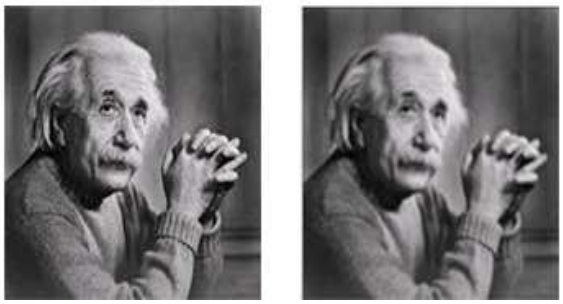

Fig -4: Blurring

Blurring helps in removing noise from an image. A low pass filter is applied to the image which allows a low change in pixel values and does not allow a high change in pixel values. Hence, significantly reducing the noise in the image.

- **Thresholding:**

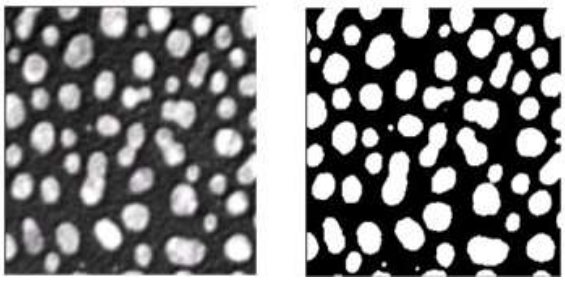

Fig -5: Thresholding

Thresholding is a form of image segmentation. A binary image is formed as a result of thresholding. All the pixels are converted to black or white depending on the threshold value.

- **Perspective Transformation**

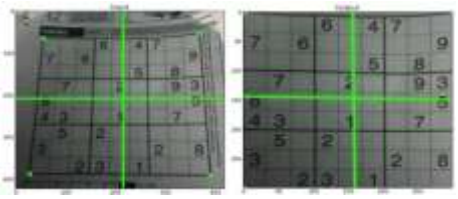

Fig -6: Prespective Transformation

Perspective Transformation is used to scale a 3- D object into a 2-D image. It is done to get a proper view of the maze.

3.2 A* Search Algorithm

A* search algorithm [7] is a technique which is used for finding the shortest path while traversing a graph. A* is a graph searching [8] algorithm which searches a huge area of the map. It is a combination of Dijkstra's algorithm and Greedy Best-first Search. It is like Dijkstra's in the aspect that it can be used to find a shortest path. It's like Greedy Best-First-Search in the aspect that it can use a heuristic. It is used to find the shortest path from the start node to the goal node having the minimum cost. It calculates the value at each node and then expands at the node with the shortest path. It continues to branch out till the goal node is reached. On reaching the goal node it traces back its path to the start node obtaining the shortest path in doing so. It minimises the formula:

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

Where, n is the next node on the path
$g(n)$ is the cost of the path from the start node to n
$h(n)$ is a heuristic function that estimates the cost of the cheapest path from n to the goal node.

A* terminates if it find the shortest possible solution to the goal or when no possible solution can be found.

3.3 8-Chain Code

The system then converts the path traced to an 8-chain code [11] that will be used by the car to solve the maze. Chain code is a popular technique generally used in image processing.

Chain codes are formed by seeing the 4-connectivity and 8-connectivity of pixels.

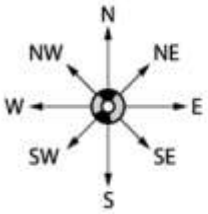

Fig -6A: Absolute 8-chain code using compass direction

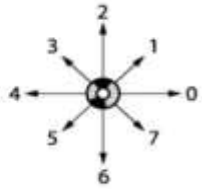

Fig -6B: Absolute 8-chain code using numbers

8-chain can be compared with the directions of a compass. The 8 points on a compass (i.e. North, North-East, East, South-East, South, South-West, West, North-West) can be assumed on an object and the object can be instructed to move in any of the above 8 directions.

We generally use numbers to write chain codes since computers can understand only binary data. Computer programmers typically use the numbered system to perform various functions. Hence a numbered system seems to be more efficient than a lettered one.

8-chain code typically is used for image compression, however we will be using a similar approach to convert our solution of the maze into simple directions to transmit to our bot.

Considering the below given maze with its best solution represented by the line. Start point and end point are represented by the red and green squares on the grid respectively:

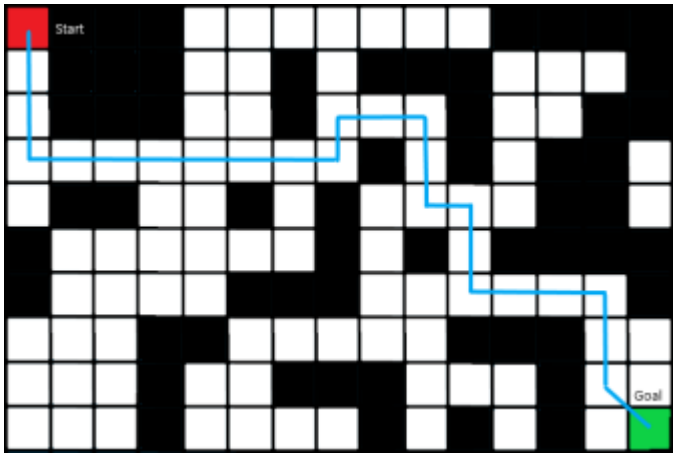

Fig -7: Absolute 8-chain code solution to a maze

The solution of the maze can be represented by the following 8 chain code:

6, 6, 6, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 6, 6,0, 6, 6, 0, 0, 0, 6, 6, 7.

The eight chain code will be used to transmit the direction in which the bot is to move in real-time. The 8-chain code breaks down the solution of the maze

into straight lines that could have 8 different directions. These directions are represented by numbers from 0 to 7. These numbers representing directions are transmitted to the bot for a specific duration of time corresponding to the length of the line. Each line is transmitted to the bot sequentially in this manner to enable the bot to traverse the maze as soon as it receives the solution. This allows the bot to solve the maze without it needing to store any data of the maze. These signals will be transmitted using the Bluetooth HC-05 module.

3.4 Transmission of data and solving the maze

We selected Bluetooth as the mode of transmission as it is wireless giving the robot freedom to move over a limited range. Bluetooth was also selected as it is easier to implement as compared other modes of wireless communication. The system then converts the path traced to an 8-chain code that will be used by the car to solve the maze. This 8-chain code will be in the form of numbered directions as explained in the preceding sub topic. This list of 8-chain code generated will thus provide a solution of the maze for the robot to follow. Each distinct value in the list tells the robot the direction in which it has to move for a given unit of time. Following this list enables the robot to solve the maze.

3.5 Bot

The robot we selected to build is an omni-directional vehicle [9], thus it can be maneuvered in 8 different direction without it needing to turn to do so. It is also capable of rotating across the vertical axis in both clockwise as well as anti-clockwise directions.

Our implementation will require the robot to move in 8 different directions only. Having an omni-directional robot eliminates the need for the robot to turn as well as it can precisely follow the path laid out by the solution.

4. DESIGN OF MAZE SOLVING ROBOT USING A* SEARCH ALGORITHM

4.1 Sequence Diagram

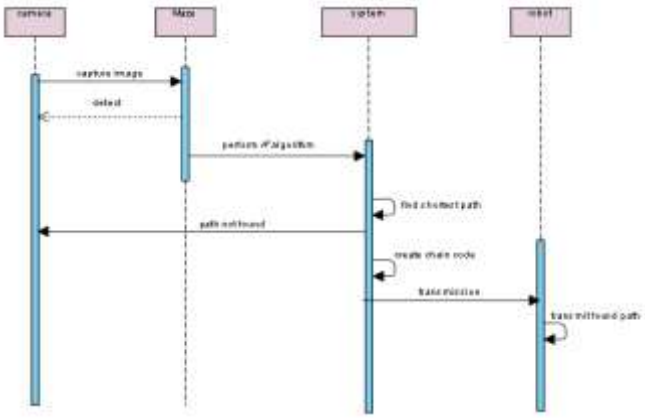

Fig -8: Sequence Diagram

The sequence of actions followed by the Maze Solving Bot is shown in the above sequence diagram. It shows the interaction between various objects of the system like the camera, maze, system and the robot. It also shows the actions performed by each object in the system. The camera will capture the image. Then the system performs the A* Algorithm on the maze and finally the robot will traverse the maze.

4.2 Block Diagram

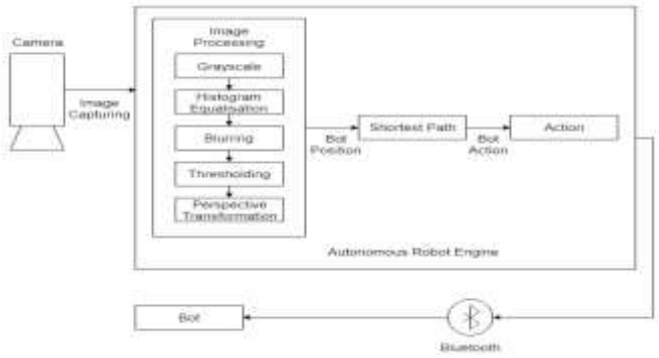

Fig -9: Block Diagram

First the image is captured using a mobile camera/webcam. The image is then sent to a computer where it undergoes various image processing operations of grayscaling, histogram equalization, blurring, thresholding and perspective transformation.

In the next step the shortest path is found using the A* search algorithm, after which it is converted into an 8-chain code and transmitted to the bot via Bluetooth.

5. SYSTEM REQUIREMENTS

The various hardware and software requirements of the system will be as follows:

5.1 Hardware requirements

- **Arduino Uno x1**:- An Arduino Uno is based on theATmega328P microcontroller. It has been developed by Arduino.cc and is an open source project. It can be programmed using the Arduino IDE (Integrated Development Environment). An Arduino Uno was the preferred choice for this task as it can easily be interfaced with its expansion boards i.e shields. The input/output pins available also gives us an option to use bluetooth/WiFi to send and receive data to the robot. Data communication can take place in analog or digital form. Programs can be loaded to the board via the USB type B connector present on the board. It can easily be powered using a 9 volt battery.
- **HC05 bluetooth module x1:-** HC-05 Bluetooth module enables us to establishes wireless

communication link with another device. It can work in either master or slave configuration. This enables the bot to receive directions to solve the maze.

- **Motor driver arduino shield L293D x1:-** The motor driver shield enables us to drive 4 DC motors independently, this is essential to be able to manoeuvre an omni-directional robot. It can also control 2 servo motors. The motor shield uses 2 L293D chips. Each is a dual-channel H-Bridge motor driver and are capable of single-handedly driving 2 DC motors or a single stepper motor. It can deliver up to 0.6A to each motor. The shield also contains a 74HC595 shift register.
- **Motors(9V 300rpm) x4:-** The motors will be attached to each wheel of the bot and each driven independently by the motor driver shield.
- **12V Li-Po Battery x1:-** It will be used to power the motor driver shield as well as the Arduino itself.
- **Omni wheels x4:-** These wheels will enable the bot to move in 8 different directions and rotate both clockwise as well as anticlockwise without the need of a steering mechanism. The independent motor control along with the omni wheels attached to the motors enables this movement by the bot.
- **Chassis:-** A chasis made of aluminium/steel will be used to mount all the hardware and components of the robot.

5.2 Software requirements

- **Arduino IDE:-** Arduino IDE is an open source software that can be used to write and upload code to the arduino. It works on Mac OS X, Windows as well as Linux. It is compatible with any Arduino board. It will be used to receive the transmitted signals and convert them to instructions that the bot will follow.
- **Python:-** Python libraries such as OpenCV and Numpy will be used to perform image processing, as well as implementing the A* search algorithm. It will be used to convert the path into an 8-chain code and transmit it to the arduino via Bluetooth using a python module such as PyBluez.

6. CONCLUSIONS

Various image processing techniques like histogram equalization, grayscale, thresholding, blurring, morphological transformation and perspective transformation can be applied on various maze images to obtain an image which can be used as an input for the A* algorithm. A* Algorithm can be used to obtain the shortest path through the maze between the start node and goal node.

Additionally, other transmission methods can be tested out to find a more efficient source that will work in a long range and also won't be blocked by obstacles like WiFi or Low Frequency Radio Waves.

For better operation, we could use sensors in the bot. It can also be transformed into a rescue bot by adding various heat sensors to detect a person and also add actuators to help perform various operations.

REFERENCES

[1] Michael Gims, "Micromouse - Microprocessor Controlled Vehicle," Bachelor of Engineering, University of East, S. L. D. B. 1999.

[2] Y. C. Chang, "Micromouse Maze Solving Robot," Bachelor of Engineering, Universiti Teknologi, Malaysia, 2009.

[3] Mohamed Alsubaie, "Algorithms for Maze Solving Robot" Manchester Metropolitan University, unit code64ET3590, 2013, pp. 12-15.

[4] Musfiqur Rahman, "Autonomous Maze Solving Robot" Department of Electrical and Electronic Engineering (EEE), University of Liberal Arts Bangladesh (ULAB).

[5] A Fast and Shorter Path Finding Method for Maze Images by Image Processing Techniques and Graph Theory
http://www.joig.org/uploadfile/2014/0516/20140516035349809.pdf

[6] Omkar Kathe, Apoorv Jagtap, Varsha Turkar, Girish Gidaye, "Maze Solving Robot using Image Processing," 2015 IEEE Bombay chapter Symposium (IBSS)

[7] Malkit Singh, Rajnish Kumar, Vaibhav Giradkar, Pallavi Bhole, Minu Kumari, "Artificially Intelligent Maze Solver Robot," International Research Journal of Engineering and Technology (IRJET),2016.

[8] Navin Kumar1, Sandeep Kaur2, "Graph Search Techniques for Finding Shortest Path in Image Based Maze Problem," International Research Journal of Engineering and Technology (IRJET), 2019.

[9] Helder P. Oliveira, Armando J. Sousa, A. Paulo Moreira and Paulo J. Costa, "Dynamical Models For Omni-Directional Robots With 3 and 4 Wheels," Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias s/n 4200-465, Porto, Portugal.

[10] Raptor B - A robot for the RoboCup Rescue Maze competition
https://italolelis.com/posts/maze-solving/

[11] Introduction to Chain Codes
http://www.mind.ilstu.edu/curriculum/chaincodesintro/chaincodesintro.php

[12] Digital Image Processing
https://www.tutorialspoint.com/dip/index.htm