Map Building, Position Tracking, localization, and Path Planning for Autonomous Intelligent Vehicles

Apoorv Waghmare[1], Chinmay More[2], Atharva Vakharkar[3]

[1]MIT-WPU, Dept of Electronics and Communication Engineering, Maharashtra, India
[2]MIT-WPU, Dept of Electronics and Communication Engineering, Maharashtra, India
[3]MIT-WPU, Dept of Electronics and Communication Engineering, Maharashtra, India

---***---

Abstract - *Autonomous Intelligent Vehicles (AIVs) can operate autonomously in a dynamic environment alongside people. For an AIV to be truly autonomous it should be able to prepare an accurate map of the area where it is supposed to function and localize itself in that map. We have developed an AIV that can build the map, localize itself in the map and track any movement that it makes. Unlike most of the AIVs, our AIV doesn't need any external guidance, this makes the vehicle truly independent. We have developed three novel algorithms that are used by the AIV for map building and map updating process and position tracking of the AIV. Using pure geometry, we have developed the Map Building and Grid Fitting (MBGF) algorithm to make an outline map of the area where AIV is supposed to function. To update the position as AIV moves, we have developed a Dynamic Map Building (DMB) algorithm. Leveraging the technology of lidar sensors and motor encoders to localize the vehicle and track each and every movement of the vehicle, we have developed the Runtime Position Tracking (RPT) algorithm. We have integrated the A-star algorithm into our system for path planning. The vehicle makes the map of the area with minimum human assistance, doesn't need any external feedback to perform the operation of localization, movement tracking, or path planning, and keeps updating the position of obstacles present in the map. Through extensive testing we have proved that all the algorithms put together work seamlessly with each other, making the AIV truly autonomous and intelligent.*

***Key Words*: Autonomous Intelligent Vehicles,** AIV, **Cartography, Motor encoders, Lidar, Localization**, etc

1. INTRODUCTION

There is an increasing need for vehicles that can transport goods or people from one place to another, especially in an indoor environment. E-Commerce companies like Amazon, Flipkart, Alibaba, etc have huge storehouses and have become dependent on small indoor vehicles for transporting goods in these massive storehouses. Earlier these vehicles were known as autonomous guided vehicles (AGVs) as these vehicles were dependent on external sensors or paths drawn on the floor to guide them from one point to another. Due to recent advancements in automation, robotics, vision, and sensor technologies, these vehicles have become more and more intelligent. The need for an external guidance system for such vehicles is decreasing day by day. The variety of applications of AIVs are also increasing. AIVs are used in the form of autonomous forklifts that are used to transport heavy goods. Huge AIVs are used toreplace assembly lines in the automobile industry where the chassis is mounted on the AIV and the AIV takes the chassis from place to place. Such AIVs are also used at parking lots to automate the process of valet parking. The whole system including sensors and algorithms can be calibrated and used for many of such AIVs. With three novel algorithms, we presenta full start to end tested solution that can be used for AVs.

2. LITERATURE REVIEW

For the effective design of any robotic system, an understanding of existing technologies is essential. For the creation of a mobile robot mechanical architecture, sensor technologies, and navigation and tracking control techniques are crucial. We looked at techniques that are used for vehicle tracking, map making, path planning, and localization in this literature review.

An autonomous intelligent vehicle can navigate itself from a starting point to a predefined destination in[1]. This document provides detailed information on the navigationof AIVs in an indoor environment.

[2]This document explains the use of lidar sensors in autonomous vehicles and how to localize the vehicle. The author concentrated on three methods for estimating a vehicle's location. The first reliable standstill detection using only signals from an Inertial Measurement Unit. Second, statistical filtering is used to estimate the state of the vehicle. Third, a LiDAR-based positioning approach that provides position, and orientation correction data.

To make appropriate driving decisions, autonomous mobile robots perceive their surrounding environment. The surrounding environment of autonomous cars is sensed using a combination of sensors such as LiDAR, radar, ultrasonic sensors, and cameras. Various physical properties of the surroundings are simultaneously captured by these heterogeneous sensors. [3]This paper shows a particular approach to making an autonomous

LiDARs are presently used in a variety of fields, including robotics, agriculture, archaeology, and the quantification of various atmospheric components. The present manuscripts cover the operation of LiDAR, its various varieties, history, and various applications. The distance between distinct objects in space may be calculated using LiDAR data, and a 3D digital representation of the scene in front of LiDAR can be drawn. [4]This paper includes a detailed performance analysis, as well as a description of the optimal lidar, the target identification algorithm, and the investigation of the lidar data accuracy and improvements using lidar-specific ground control targets in the case of various target types.

Autonomous Intelligent vehicles are frequently used to testa wide range of algorithms and to simulate real-world driving behavior. [5]This research employs a Raspberry Pi and a LIDAR module designed exclusively for interior navigation to demonstrate autonomous vehicle deployment that effectively addresses this difficulty. We need to maintain a dataset if we utilize LIDAR, and LIDAR is highly expensive when compared to IMU (magnetometerand optical encoder). If these sensors are completely adjusted, the output will be sufficient for the bot to grasp its location.

The difficulty of tracking a moving robot across an environment with imperfect sensor data and knowledge ofthe vehicle's motion is known as localization. [6]This paper demonstrates one of the ways for solving the bot's localization problem. It is a probabilistic-based strategy for resolving the challenges of localization. The robot's belief is represented by a set of weighted samples and modified according to motion and sensor inputs using a Bayesian formulation of the problem. It is resistant to mapping mistakes, which inevitably reduce the accuracy of the results. This function is used to dynamically update the

mobile robot which is using a LIDAR and a wide range camera for free space detection and shows how to align the output of these sensors using a geometrical model to map the area of functioning of AIV.

Map during the localization process without necessitating a significant increase in running time

Vehicles based on LIDAR sensor-measured point clouds [7] When only partial scanning data is available, this work solves the problem. This work provides a non-shape-modeling solution to the obstacle categorization problem for partial point clouds. The method is validated against a known database as well as real-world settings. In the case of AGVs, the real-time operation is supplied by standard-complexity onboard computers.

An encoded motor helps us to acquire the RPM of the motor which on further processing can help us track the bot's current location [8]This paper describes how encoded motors can help in-vehicle localization. The current study explains how to conduct a real-time velocity analysis of a DC motor using an encoder.

An efficient algorithm for map generation is crucial for the path planning of an AIV. [9] This work investigates the creation of local elevation maps and autonomous motion planning for unmanned ground vehicles based on possible direction angles. An algorithm for constructing a local mapbased on numerous lasers is introduced, based on the configuration of a sensor system mounted on an unmanned ground vehicle, and the influence of dynamic obstacles is eliminated by combining with an algorithm fortracking dynamic obstacles.

We have used a LIDAR sensor to generate a top view outline map of the designated area of operation. [10]The wheelchair navigates by using the ROS operating system, microcontrollers, rotary encoders, and a LIDAR device. The LIDAR unit takes important environmental measures that are utilized to create a map of the area. The LIDAR sensor and the rotary encoders are used by ROS to determine a passable path to a destination defined by the user. To propel the wheelchair along the chosen path, ROS sends movement orders to a microcontroller. The navigation system adapts to changes in the environment and reroutesthe wheelchair if new obstacles appear.

3. ELECTRONIC COMPONENTS

1.1 Raspberry Pi 3b+

Fig-1: Raspberry Pi 3b+

Raspberry Pi 3B+ is a powerful microprocessor that runs aRaspian operating system on it. The Rapspian operating system is a lite version of Linux. Raspberry Pi is the onboard computer. It acquires data from various sensors present on the AIV, does all the processing that is needed, and controls the motion of AIV by controlling both motors.

1.2 Proximity Sensor HC-SR04

Fig-2: Proximity sensor HC-SR04

HC-SR04 is an ultrasonic proximity sensor. This sensor is used to acquire the distance of an obstaclepresent in front of this sensor.

1.3 SPG30E-60K DC Geared Motor

Fig-3: SPG30E-60K DC motor

SPG30E-60K 12 volt DC geared motor equips quadrature hall effect encoder at the backside of the motor. The sensor produces 810 counts per main shaft revolution, 3 pulses per rear shaft revolution, single-channel output. This sensor is used for measuring the rpm of the motor.

1.4 RP-Lidar A2M7

Fig-4: RPLIDAR A2M7

RPLIDAR A2M7 is a 360-degree type of lidar sensor. It has a 16m scanning radius and its sampling rate can go as high as 6000 samples per second. This Lidar sensor's core rotates clockwise to perform 360-degree omnidirectional laser range scanning for its surroundings, then generates 2D point cloud data of a cross-section of the environment.

1.5 Lipo Battery

An 11.1 volt, 2200mah Lipo battery is used to power both the motors of the AIV.

4. MECHANICAL ARCHITECTURE OF AIV

Fig.5 represents the CAD model that we designed for our AIV and Fig.6 is the image of the actual AIV that we have manufactured.

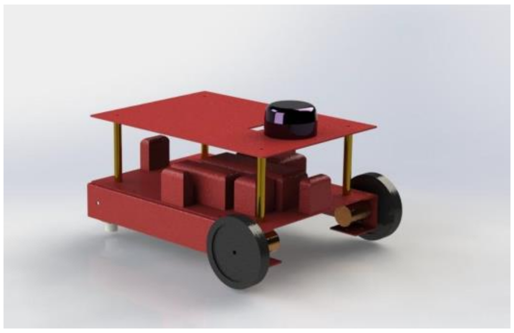

Fig-5: CAD model of AIV we have developed

Fig-6: AIV we have developed

Considering the stability, maneuverability, and sensor positioning the design of the AIV was finalized. The AIV has a rectangular double-decker chassis. The AIV has a four-wheel design. The rear two wheels are castor wheels that can move freely in any direction. The front two wheels are attached to motors. Front wheels maneuverability the AIV. This type of wheel arrangement allows the AIV to have the turning radius of any value including zero (turning radius zero means the bot rotates in place).

5. MAP BUILDING AND GRID FITTING PROCESS

The AIV has to know the physical workspace where it will function, that is the AIV should know the dimensions of the perimeter of the workspace. We are proposing this novel algorithm which can be used to generate the outline map of the workspace and put a grid on this workspace. This grid is further used for path planning and localizing the AIV.

The four stages of this MBGF are mentioned below in detail:

1st Stage: Considering any one point on the perimeter of the actual area as (0, 0) i.e. origin of a cartesian plane the user has to measure coordinates of all the vertices of the area. If the perimeter is curved somewhere then we can consider a few equidistant points on that curve, which can produce a good approximation of the curve.

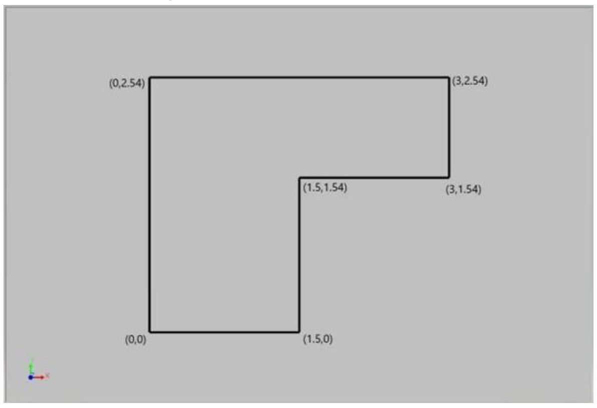

Fig-7: Map polygon

**{ (0,0) , (1.5,0) , (1.5,1.54) , (3,1.54) , (3,2.54) , (0,2.54)
}**

The above set of coordinates are fed to our algorithm. Our Algorithm fits a polygon in this set of coordinates. Then this polygon becomes the outline of the workspace where AIV has to function. We call this polygon "map polygon". This 1st stage is a one-time process.

2nd Stage:

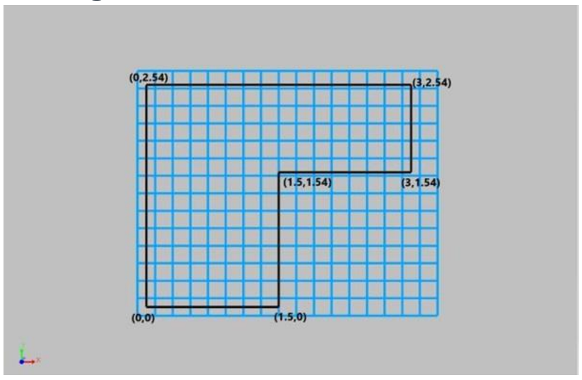

Fig-8: Full grid and map polygon

In the 2nd stage, we put a grid on the map polygon. All the grid cells of this grid are square and are of equal size. The grid cell size is to be decided by considering the size of the AIV and the size of the map polygon. We call this grid "full grid".

3rd Stage:

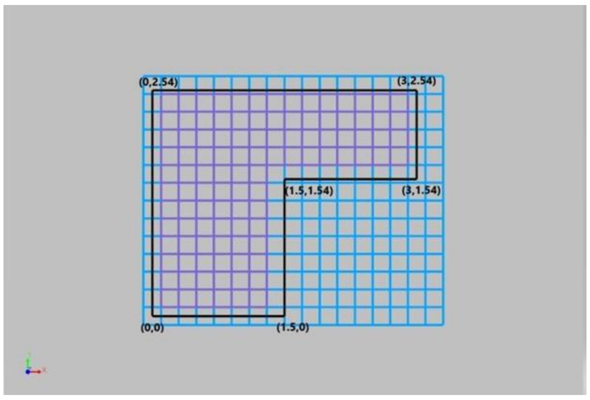

Fig-9: Valid grid, full grid, and map polygon

In the next stage, we only consider the grid cells lying completely inside the polygon. Those who are outside the polygon are eliminated. The grid cells whose edges overlap with the edge of the map polygon are also eliminated, this is done for safety purposes. Here as we can see only the purple-colored cells are valid. We call this grid a "valid grid".

4th Stage:

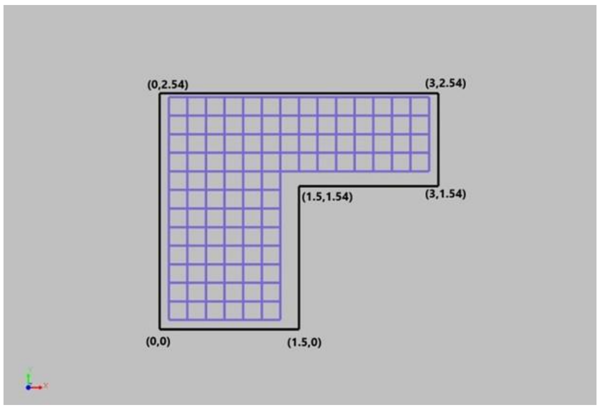

Fig-10: Valid grid and map polygon

We have eliminated the invalid grid cells and we will only consider the valid grid cells. This valid grid forms the workspace for the AIV. Splitting the area into grid cells of equal size is very helpful for most path planning algorithms.

6. PATH PLANNING

The AIV always knows the coordinates of its own position on the map. Using those coordinates we find out the grid cell in which the AIV lies. We call this grid cell "present grid cell". When the user feeds the destination point to AIV we find out the grid cell in which the destination point lies. We call this grid cell "target grid cell".

The A-star algorithm was selected because it is perfect for grid-based searching. As the A-star algorithm plans paths on the fly, the computation required for the A-star algorithm is significantly less than other algorithms like the Djikstra algorithm which considers all the possible paths at the beginning itself. Less computation is always better to handle on microprocessors and microcontrollers.We apply the A-star search algorithm to do grid cell-based searching of the path from the present grid cell to the target grid cell. Joining the centers of the grid cells through which the path is planned we get a polyline from the present position of the AIV to the destination point. This polyline is executed by the AIV to go from the current position to the destination point. Fig 11 represents the center points of the grid cells through which the path is planned.

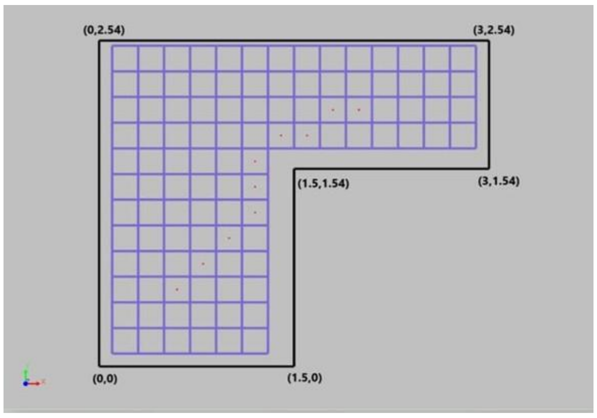

Fig-11: Planned path, valid grid, and map polygon

7. POLYLINE EXECUTION

The path planned by the AIV is a polyline. The algorithm that we have developed to execute the polyline requires the AIV to know its current position coordinates and the direction vector at every given point of time. The direction vector of the AIV is the direction towards which the front part of the AIV is pointing. The direction vector of the AIV is always a unit vector. The AIV executes the planned polyline segment by segment.

Following are the steps for execution of a segment of a polyline:

- AIV knows its current position coordinates, direction vector, and coordinates of the next pointin the polyline.
- AIV calculates the vector between current position coordinates and the next point in the polyline. This vector is then normalized to get a unit vector.The unit vector in the direction vector along which the AIV should travel to get to the next point in the polyline. By calculating the dot product between the two vectors the AIV calculates the angle between the current direction vector and the expected direction vector. If the angle lies between 0º to 180º the AIV rotates its left wheel backward and right wheel forward to rotate left in place. If the angle lies between 0º to -180º (-108º included) the AIV rotates its left wheel forwards and right wheel backward to rotate right in place. When the dot product between the current direction vector and the expected direction is between -0.95 and +0.95 the AIV stops rotating.
- Once the expected direction vector and current direction vectors are aligned the AIV calculates the distance between current position coordinatesand coordinates of the next point in the polyline.

AIV travels this distance in a straight line.

The above process is executed for all the line segments of the polyline. This is how the AIV executes the planned polyline from its current position to the destination.

7. RUNTIME POSITION TRACKING ALGORITHM

To execute any motion (in our case polyline motion) the AIV has to know its position coordinates and direction vector at each point in time. We propose this novel algorithm with a detailed mathematical explanation to track position coordinates and direction vector of AIV. This algorithm uses motor encoders to measure the rpm of both the motors independently and uses this rpm to trackcoordinates and direction vector of the AIV.

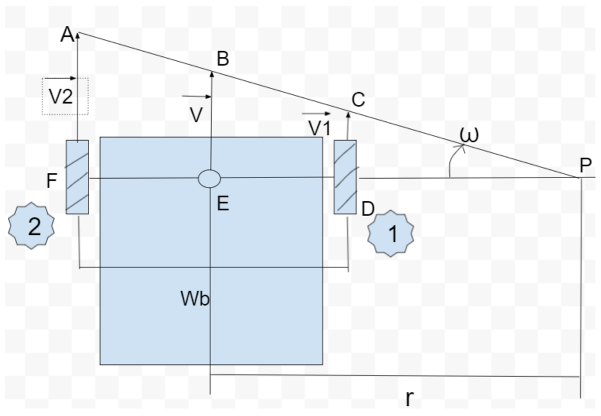

Fig-12: Velocity vectors and instantaneous point of rotation

Fig 12 is the top view of our AIV. The right side wheel is wheel 1 and the left side wheel is wheel 2.

Where,

$\overline{v1}$: Velocity Vector of wheel 1

$\overline{v2}$: Velocity Vector of wheel 2

P: Instantaneous point of rotation of AIV

E: Centre point between two wheels. Coordinates of thispoint are considered as coordinates of AIV.

r: Radius of the instantaneous circle of rotation.

: angular velocity about P

$$ED = \frac{wb}{2}$$

$$r = DP + \frac{wb}{2}$$

By the law of Similar Triangles,

$$\frac{FP}{DP} = \left|\frac{\overline{v2}}{\overline{v1}}\right|$$

$$\frac{wb + DP}{DP} = \left|\frac{\overline{v2}}{\overline{v1}}\right|$$

$$|\overline{v1}|wb + |\overline{v1}|DP = DP|\overline{v2}|$$

$$|\overline{v1}|wb = DP(|\overline{v2}| - |\overline{v1}|)$$

$$DP = \frac{|\overline{v1}|wb}{|\overline{v2}| - |\overline{v1}|}$$

As per equation (1),

$$r = \frac{|\overline{v1}|wb}{|\overline{v2}| - |\overline{v1}|} + \frac{wb}{2}$$

$$r = \frac{2|\overline{v1}|wb + wb(|\overline{v2}| - |\overline{v1}|)}{2(|\overline{v2}| - |\overline{v1}|)}$$

$r = \frac{wb(|\overline{v1}| + |\overline{v2}|)}{2(|\overline{v2}| - |\overline{v1}|)}$...(radius of instantaneous circle of rotation)

As per the law of similar triangles,

$$|\overline{v}| = \frac{|\overline{v1}| + |\overline{v2}|}{2}$$

the magnitude of the instantaneous velocity vector of the AIV is calculated.

To find the direction of this velocity vector.

$= \frac{|\overline{v2}| - |\overline{v1}|}{wb}$ ... (as $v =$ $R\, formula$)

$$= \frac{d\theta}{dt}$$

Therefore, $d\theta = \left(\frac{|\overline{v2}| - |\overline{v1}|}{wb}\right)dt$ ... (2)

wb: Wheel Base

$r = DP + ED$ ...(1)
$vi = ai\ i + bij$
be the initial velocity vector.

Where, ' ' is angular velocity about point 'P'.

Let

The previous velocity vector is rotated by '$d\theta$' to get thedirection of the current velocity vector.

Let $\overline{vc} = ac\ \hat{\hat{i}} + bcj$ be the current velocity vector.

$$\begin{bmatrix} ac \\ bc \end{bmatrix} = \begin{bmatrix} cos(d\Theta) - sin(d\Theta) \\ sin(d\Theta) cos(d\Theta)) \end{bmatrix} \begin{bmatrix} ai \\ bi \end{bmatrix}$$

$$\begin{bmatrix} ac \\ bc \end{bmatrix} = \begin{bmatrix} 1 & -d\Theta \\ d\Theta & 1 \end{bmatrix} \begin{bmatrix} ai \\ bi \end{bmatrix}$$

$ac = ai - bi(d\theta)$, $bc = ai(d\theta) + bi$

$ac = ai - bi(\frac{|\overline{v2}| - |\overline{v1}|}{wb})dt$

$bc = ai(\frac{|\overline{v2}| - |\overline{v1}|}{wb})dt + bi$ ... as per equation 2.So

the direction of the velocity vector is,
$\overline{vc} = ac\ \hat{i} + bc\ \hat{j}$

The current velocity vector is normalized to get a unit vector along the direction of the current velocity vector. Then we have to multiply it with the magnitude of the current velocity vector that we found earlier. This is how we get the instantaneous velocity vector of AIV.
This algorithm only requires encoded motors and no othersensors for the runtime tracking of the vehicle. The mathematics that is formulated is generic for all two-wheel drive AIVs. Using this algorithm we can constantly keep track of any movement that the vehicle makes.
This algorithm runs in a separate thread on raspberry pi. The decision for running this algorithm in a separate thread was taken because it helps to isolate the tracking process from other processes running on raspberry pi which reduces the lag and more accurate tracking is done. As the tracking process is constantly running it will not miss time frames where the vehicle is moving.

transformation matrix to localize the AIV and find its direction vector. This algorithm is used when AIV is initialized to find its location on the map. This algorithm is also used to correct the position coordinates and direction vector from time to time. We cannot keep on doing this process throughout the runtime because it requires a lot of time and the AIV has to be completely stationary when point cloud 1 is generated. The RPT algorithm and localization using lidar work hand in hand to produce a fully independent localization and tracking system for AIV.

8. USE OF LIDAR SENSOR FOR LOCALIZATION

The runtime position tracking using motor encoders cannot determine the coordinates of the position when the AIV is booted, and will not be able to eliminate cumulative errors. For example, the errors that are generated due to mechanical defects. To solve these two problems we are using a lidar sensor. Lidar sensor generates a 2D point cloud of a cross-section of the map, let's call this point cloud 1. We register this point cloud data to the point cloud data of the full map, let's call this point cloud of the map. When we register this point cloud 1 to point cloud of map we find out the transformation matrix for point cloud 1 with respect to point cloud of the map. We use this

8. DYNAMIC MAP BUILDING AND MAP UPDATING ALGORITHM

The map consists of a grid and the grid is made up of multiple grid cells. But in the MBGF algorithm, we have not taken care of the obstacles present inside the workspace. To take care of the obstacles present in the map we have developed this algorithm. This algorithm uses ultrasonic proximity sensors to detect obstacles.

Each grid cell has a status assigned to it. The status can be active or inactive. Grid cells having no obstacle present in them are active. Grid cells having obstacles present in them are inactive. Each grid cell gets three penalty chances (penalty 0, penalty 1, penalty 2, and penalty 3). By default, each grid cell in the valid grid is active and has a penalty 0.

Following are the flow charts of dynamic map building andmap updating algorithm:

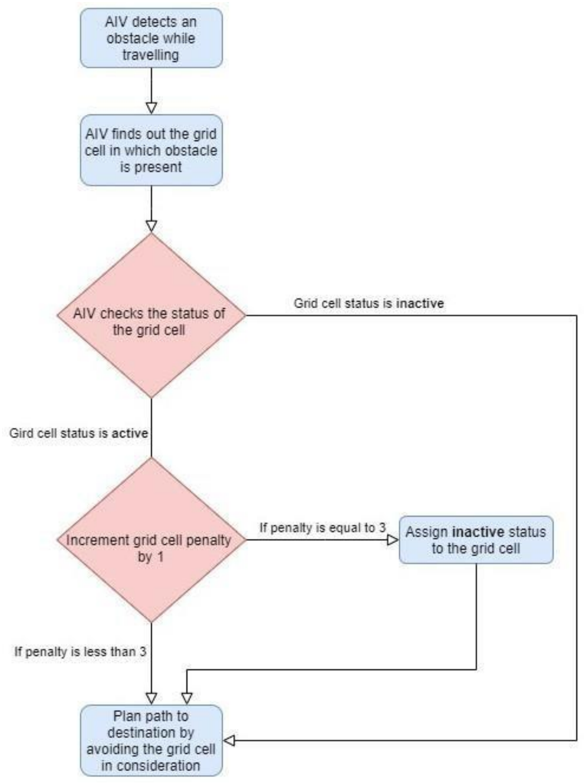
AIV detects an obstacle while travelling
AIV finds out the grid cell in which obstacle is present
AIV checks the status of the grid cell
Grid cell status is inactive
Gird cell status is active
Increment grid cell penalty by 1
If penalty is equal to 3
Assign inactive status to the grid cell
If penalty is less than 3
Plan path to destination by avoiding the grid cell in consideration

Fig-13: Algorithm for updating obstacles present in the area

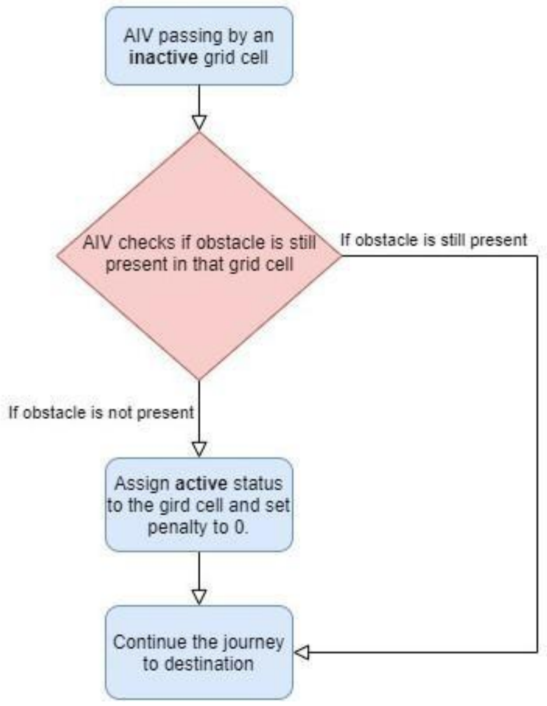
AIV passing by an inactive grid cell
AIV checks if obstacle is still present in that grid cell
If obstacle is still present
If obstacle is not present
Assign active status to the gird cell and set penalty to 0.
Continue the journey to destination

Fig-14: Algorithm for updating the obstacles that are removed from the area

This algorithm will help to keep the map up-to-date. The more the AIV travels the more up-to-date the map is. If an obstacle is introduced in the area it will be reflected in the map AIV has generated. If any obstacle is removed it will also be reflected on the map. This algorithm requires very little computation and uses only proximity sensors, which makes it perfect to run on microprocessors and microcontrollers. This algorithm is very easy to implement. Through extensive testing, we have concluded that this algorithm is perfect for continuously updating themap of a dynamic environment.

CONCLUSION

In this paper, we put forward a detailed description of a full-fledged system for AIV that we have developed and tested. The MBGF algorithm, RPT algorithm along with lidar, A-star algorithm, and DMBMU algorithm put together work seamlessly to make a vehicle truly autonomous and intelligent. No matter what the AIV is made for, the map-making and tracking process is mandatory for each AIV. The system that we have developed can be calibrated and used for AIV of any shape and size.

REFERENCES

[1] E. S. Morales, M. Botsch, B. Huber and A. G. Higuera, "High Precision Indoor Navigation for Autonomous Vehicles," 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Pisa, Italy, 2019, pp. 1-8, doi: 10.1109/IPIN.2019.8911780.

[2] De Silva, V.; Roche, J.; Kondoz, A. Robust Fusion of LiDAR and Wide-Angle Camera Data for Autonomous Mobile Robots. Sensors 2018, 18(8), 2730; https://doi.org/10.3390/s18082730.

[3] Papoutsidakis, Michail & Kalovrektis, Konstantinos & Drosos, Christos & Stamoulis, Georgios. (2017). Design of an Autonomous Robotic Vehicle for Area Mapping and Remote Monitoring. International Journal of Computer Applications. 167. 36-41. 10.5120/ijca2017914496.

[4] https://www.ingentaconnect.com/content/asprs/p er s/2007/00000073/00000004/art00004;jsessionid =4 cam8541dpaun.x-ic-live-01 Number 4 / April 2007, pp. 385-396(12) Csanyi, Nora; Toth,CharlesK.

[5] N. Baras, G. Nantzios, D. Ziouzios and M. Dasygenis, "Autonomous Obstacle Avoidance Vehicle Using LIDAR and an Embedded System," 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, Greece, 2019, pp. 1-4, doi: 10.1109/MOCAST.2019.8742065.

[6] Milstein, Adam. (2005). Dynamic Maps in Monte Carlo Localization. 1-12. 10.1007/11424918_1.

[7] Z. Rozsa and T. Sziranyi, "Obstacle Prediction for Automated Guided Vehicles Based on Point Clouds Measured by a Tilted LIDAR Sensor," in IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 8, pp. 2708-2720, Aug. 2018, doi: 10.1109/TITS.2018.2790264.

[8]]Parikh, Priyam, et al. “Velocity Analysis of a DC Brushed Encoder Motor using Ziegler-Nichols Algorithm: A Case of an Automated Guided Vehicle.” Indian Journal of Science and Technology, vol. Vol 9(38), no. 10.17485/ijst/2016/v9i38/100884, 2016, p. 8. Researchgate.net.

[9] J. Nan, L. Yisha, Z. Yan and W. Wei, "Real-time map building and autonomous motion planning for unmanned ground vehicles," Proceedings of the 31st Chinese Control Conference, 2012, pp. 4930-4935.

[10] H. Grewal, A. Matthews, R. Tea and K. George, "LIDAR-based autonomous wheelchair," 2017 IEEE Sensors Applications Symposium (SAS), 2017, pp. 1-6, doi: 10.1109/SAS.2017.789408