Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 03, March-2016

A Hybrid Combination of Navigation and Path Selection Algorithm for Controlling AGV

J.Sankari, Sai Saraswathi,
ECE Department,
Sri Manakula Vinayagar Engineering College,
Puducherry, India

R. Imtiaz
Assistant Professor, ECE Department,
Sri Manakula Vinayagar Engineering College,
Puducherry, India

***Abstract*— Automated Guided Vehicle (AGV) is the future drift that provides unmanned transportation - that transports all kind of products without human intervention within production, logistic, warehouse and distribution environments. In existing work, the input given to the system is the whole map of the sector which needs larger memory along with increase in execution time. In order to overcome these limitations the proposed work has a key feature of AGV in which they can operate as a standalone system with higher efficiency. The proposed work has a predominant feature of AGV that is being implemented using Multi-storage path algorithm for navigation and A* algorithm used for path selection in case of an obstacle on its navigation path. The A* algorithm is considered as its execution time is much faster than the previous algorithm for selecting an alternative path. The structural features are emphasized using Gazebo simulator in ROS platform. The obtained Automated Guided Vehicle is being controlled using Wifi that facilitates and makes it ease of controlling over long distance.**

I. INTRODUCTION

The Automated Guided Vehicle (AGV) is used in most of the manufacturing systems for loading and unloading the materials. The main usage of AGV is for dispatching the goods from one location to another with greater ease. The AGV can tow objects behind in trailers to which they can autonomously attach. Such trailers are utilized to move raw materials or finished product. Earlier this was done completely by human effort which was not that efficient. The human work has been replaced by the usage of Line followers[1]. The line followers have been classified into major types based on their utility. Wired line follower in which a slot is being made on to the floor and a wire is placed along which the AGV follows using the sensor that detects the relative positioning of the radio signal. The main disadvantage of wired line follower is that the installation becomes very tedious in case of dynamic industries. The Wired Line followers[2] are restored by Guided Tape[3] in which the AGV's use tapes for the guide path. These tapes are of two types: magnetic or colored The AGV is fitted with the apt guide sensors that are used to follow the path of the tape. The advantage of guided tapes over wired guidance is that it can be easily removed and relocated. A flexible magnetic bar[4] can be embedded in the floor, like a wire - but works similar to the provision as magnetic tape and so remains unpowered or passive. Another benefit of magnetic guide tape is the double polarity. Small pieces of magnetic tape may be placed to change states of the AGC (Automated Guided control) based on polarity and sequence of the tags. The Guided Tape is though its initially a smaller amount, but lack the advantage of being embedded in high traffic areas in which the tape become damaged or dirty.

The next to the Guided Path its Laser target navigation[5] in which routing is made by mounting reflective tape on walls, pole or fixed equipment. The AGV carries a laser transceivers on a rotating turret using which the angle and distance to any reflectors that in line of sight are automatically calculated. This data is compared to the map of the reflector layout stored in the AGV's memory and attains the required navigation. The demerit of Laser Target navigation is that it produces greater overheads and requires more system maintenance.

Currently in the world of consumer society, where the corporations which are seeking to improve work efficiency, minimize the cost of human operators in logistics, and also bring the production cycle time down, make an accurate utilization of robots can promote the operation of the working process by simplifying it to greater extent. Such progress is rendered by Automated guided vehicle (AGV) a type of mobile vehicle that works without human intervention. Compared to the previous trends the navigation in AGVs more often than not use signal paths[6], lane paths or beacons. The AGV's are completed by the usage of sensors for navigation is optical sensors, magnetic sensors, laser scanner. Modern AGV differ from the conventional ones, instead of using fixed paths many modern AGVs are free-ranging[7]. Thus their preferred tracks are software programmed, and can be changed fairly easy when new stations or flows are supplemented. While moving the vehicles it is necessary to ensure the safety of the environment[8], personnel are taken care by the vehicle itself. The safety is ensured by the process of sensors that detect obstacles and approaching danger in their path. This paper proposes a method of AGV navigation which is constructed using ATMEGA 328 controller used for controlling and coordinating all the peripherals. The motion of AGV is obtained using 4-servo motors which are driven by 2 motor drivers. Separate EEPROM memory module is being used for storing the inputs. The AGV is controlled using a Bluetooth module. This paper is organized as follows: The AGV system description is presented in Section II. Section III explains the path re-planning in presence of any obstacle. The Simulation and Experimental analysis are shown in Section IV. The paper is summarized with a conclusion in Section V.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 03, March-2016

II. SYSTEM DESCRIPTION

The AGV constructed by ARDUINO board which is represented in Figure 1

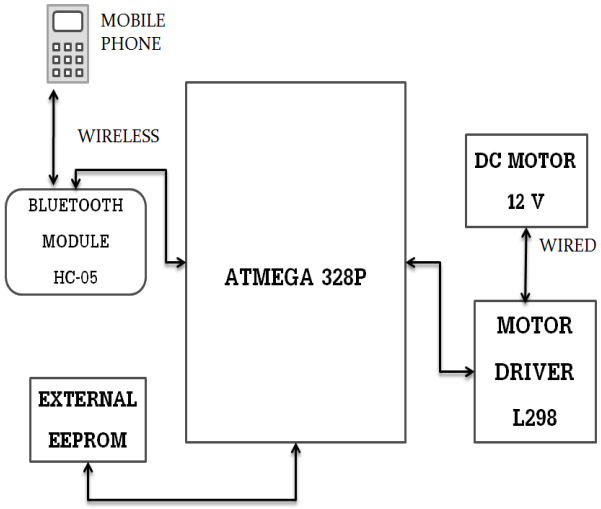


Fig1: Hardware description for proposed AGV

Atmel®AVR®Atmega 328 in proposed AGV has a low power CMOS 8-bit microcontroller based on the AVR RISC architecture. They are capable of executing powerful instructions in a single clock cycle; the Atmega328 achieves throughputs approaching 1 MIPS per MHz, which allows the system designed to optimize power consumption as well as processing speed. They posses 14 digital input/output pins in which 6 can be utilized as PWM outputs 6 analog inputs, a 16 MHz quartz crystal, a USB slot, a power jack, an ICSP header along with a reset button. They provide everything that is required to connect with a PC for inter-communication[9]. The board can be powered either using a AC-to-DC adapter or with the USB port or using a battery. The navigation of AGV is obtained using DC motors which are connected to the controller using an interface of motor driver. The L293D motor driver is used which facilitates AGV navigation with ease. The L293D pin diagram is represented in Figure 2

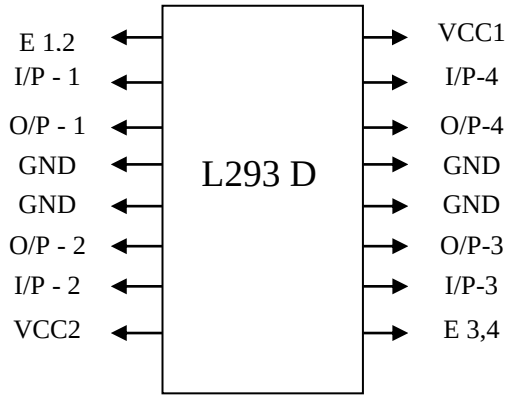


Fig 2: Pin diagram of L293D (motor driver)

The data are stored into the EEPROM module. There is a need for additional EEPROM module as the EEPROM storage in Arduino is not appreciable[10]. The data is fetched in such a way that they are executed in either ways – top to bottom or bottom to top fashion. But as it's required to execute in both ways the programming is made for dual execution. The usage of EEPROM takes a vital part in AGV. Taking up the advantage of EPROM (non volatile), the robot can retain and progress with their current execution even when the system comes back to the powered state after a period of power loss at crisis conditions.

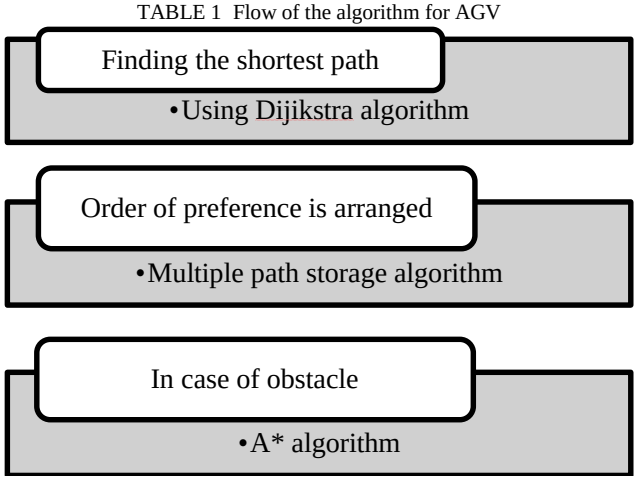
Another important feature that is used for controlling the AGV is the Bluetooth Module. The usages of mobile phones have increased to a greater extent. Thus it would be very simpler in case of controlling systems through phones. One such feature is enabled using the Bluetooth module[11]. They enable us to control the AGV in different states from our place within the working sector (ought to be within visible range of 10m). The Bluetooth module is represented in Figure 4. They posses 4 pins in which VCC – is used for powering the module, along with TX – for transmitting and RX – for receiving and along with that VSS – for grounding purpose.
All the components are arranged as shown in the Figure and thus the AGV navigation is made possible.

III. PROPOSED ALGORITHMS

A. NAVIGATION

The navigation is achieved using wireless by controlling the AGV using Bluetooth/Wifi. The Dijikstra algorithm is used so as to get the shortest distance in the given path.

TABLE 1 Flow of the algorithm for AGV



Starting point is called as initial node. Let the distance of node *Y* be the distance from the preliminary node to *Y*. Dijkstra's algorithm will allocate some initial distance values and will try to improve them step by step.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 03, March-2016

1. Allot to every node a tentative distance value put it to zero for our initial node and to perpetuity for all other nodes.
2. Put the initial node as current. Blot all other nodes unvisited. Make a set of all the unvisited nodes called the unvisited set.
3. For the current node, regard as all of its unvisited neighbors and calculate their tentative distances. Evaluate the recently calculated uncertain distance to the current assigned value and assign the smaller one. For example, if the current node *A* is marked with a space of 3, and the rim connecting it with a neighbor *B* has length 2, then the distance to *B* (through *A*) will be 3 + 2 = 5. If B was previously distinct with a distance greater than 5 then change it to 5. Otherwise, carry on with the existing value.
4. Mark the existing node as visited and take out it from the unvisited *set*. A visited node will never be checked again.
5. If the destination node has been marked visited (when setting up a way between two precise nodes) or if the least tentative distance among the nodes in the *unvisited set* is infinity (when planning a whole traversal; occurs when there is no association between the initial node and outstanding unvisited nodes), then stop. The algorithm has over and done with.
6. Or else, choose the unvisited node that is marked with the least tentative distance, set it as the new "present node", and return to step 3.

IV. PATH REPLANNING

The need for path re-planning is as in the case of any obstacles that arises in the path of the AGV. Consider a path 'A' in which the AGV is traversing, if the path is obstacle free then AGV moves without and interventions and reaches the destination as per the program. But as in case of any obstacle is seen in the path of AGV towards the destination path then there arises a problem. Such obstacles can be sensed using the ultra sonic sensors. The ultrasonic sensor is given by Figure 5. When any obstacle is detected then AGV tends to stop at that place and waits for the obstacle to move in case of any dynamic one. But when it becomes a static one then there requires a complete change of a path to reach the destination for which the path replanning is required. Its not necessary to have the complete map of the location in which the AGV is to driven. Instead Multiple path storage can be done which saves the memory. To understand the concept of multiple path storage ought to have an idea about the dry run and free run. The dry run is nothing but the first time configuration of the vehicle to a path. It is like for the first time giving the instruction to AGV so that it's capable of storing the data in it. And this dry run is done only from the source to the destination and not for the reverse process. Because the reverse gets executed automatically this doesn't require further repetition of coding because of which the memory is saved to certain extent and can be used for other purposes. After dry run the path is stored in the memory of AGV and thus the direct job of AGV to get executed when it is left in the same path 'A'.

As considered before in case of any obstacle in the path 'A' subsequently its necessary to take up a new path for the completion of job. In order to achieve that instead of feeding the complete map of the place the multiple path storage can be used. In this as an initial step the shortest path to the destination is found in the ascending order using any best shortest path algorithm. The path replanning is represented using the Figure 3

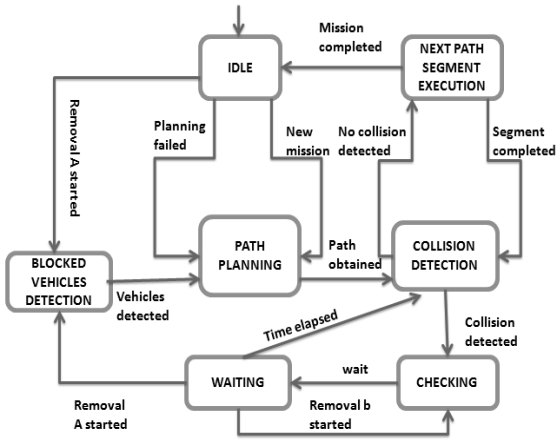


Fig 3: Path replanning

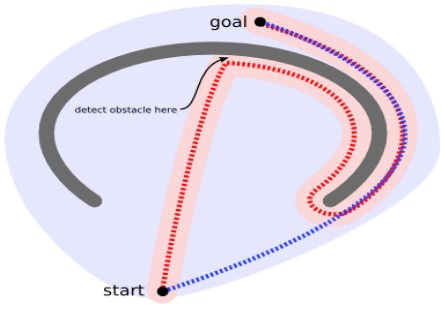
Then the dry run is made for the entire shortest path that was found using the algorithm. After which in case of any obstacle in the path 'A' which is considered to be the shortest path, then the second shortest path is taken and thus the destination is reached without any interventions. As a future work this can be extended for controlling AGV over internet in which the controlling capability is extended to a greater distance. This can be used in the case of large area.

The path selection is done using the A* algorithm. The A* algorithm is executed in such a way that it calculates the shortest path whose result is much better than the conventional Dijikstra algorithm. In A* algorithm,

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

$$h(n) \leq d(m,n) + h(m) \quad , d(m,n) \rightarrow \text{nodes}$$

The above equation is used in which h(n) is the heuristic value and g(n) is the actual distance. According to the A* algorithm in case of any contradiction between the Dijikstra algorithm and A* algorithm arises then the value of the latter is considered because of its precise result. And on that case they create a virtual obstacle on the navigation path and proceeds with the shortest distance that is given by A* algorithm.



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 03, March-2016

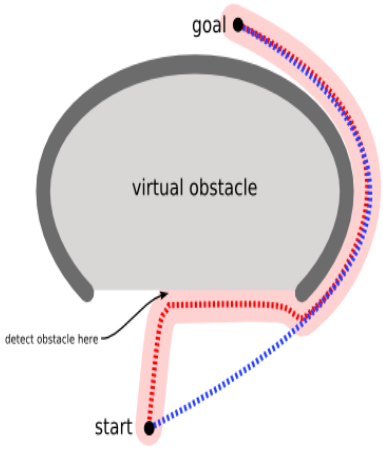


Fig 4: a) Finding the shortest path using A* algorithm , b) Creating a virtual obstacle

V. SIMULATION AND RESULTS

The simulation is obtained using two simulators on different Operating Systems. The results are obtained by executing the code for AGV in either direction. This is executed in a series manner which follows progressive steps. The input data is being given to the EEPROM for the first time. The process of feeding the input to the EEPROM is called as Dry run. The dry run means the that the data is given and the first time configuration is performed.
After the dry run process the data must be read from the EEPROM. The data fetching from the EEPROM must be performed in either ways

a) In forward direction
b) In reverse direction

The need for the forward direction is to traverse from the source to the destination in the given path using its appropriate algorithm. This literally means that the navigation of AGV is to obtained from the source to the destination (forward direction). As the forward direction execution is performed it has to be proceeded with the reverse direction. The word reverse direction refers to the traversing of AGV from destination to the source. The reverse direction execution is performed by FREE RUN. The Free run refers to the execution of the dry run in either direction. As the first time configuration (dry run) is done, the free run is performed that enables both side navigation from source to destination and vice versa.

The pictorial structure of the constructed robot is seen using the Gazebo simulator. Using the gazebo simulator the structure of the bot is obtained.

As an initial step the base of the bot is obtained in gazebo on the Ubuntu linux platform.

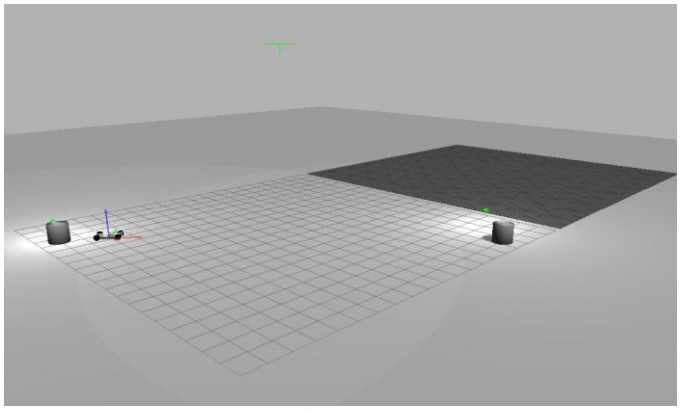


Fig 5 : Navigation from source to destination

The basic work of the constructed bot is to navigate from the source to destination. The above fig 5 shows that the bot is traversing on a basic structural arena for executing the navigation.

Secondarily, whenever an obstacle is seen on the pathway of the navigation the bot should get halted and should make over its decision to attain their next shortest distance. This is obtained by the fig 6 that takes up an alternative path in case of any obstacle on its navigation pathway. The bot that is being navigated on its initial level is free of various hurdles. Thus, the navigation path is limited by creating a maze like feature on which the navigation of AGV has to be taken from source to destination on the given arena.

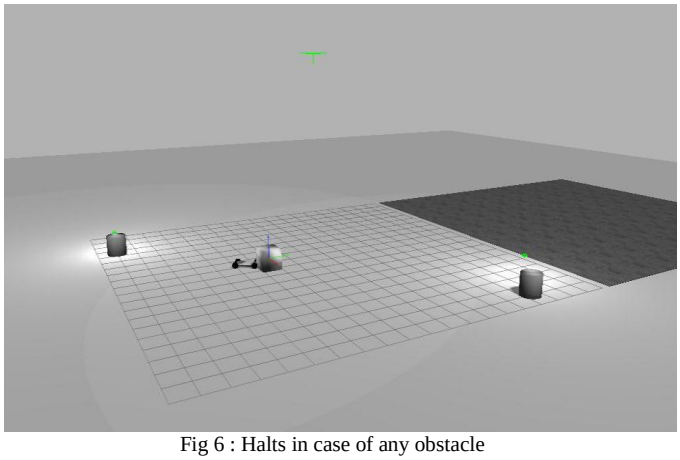


Fig 6 : Halts in case of any obstacle

The fig 7 represents the robot being executed on the give maze.

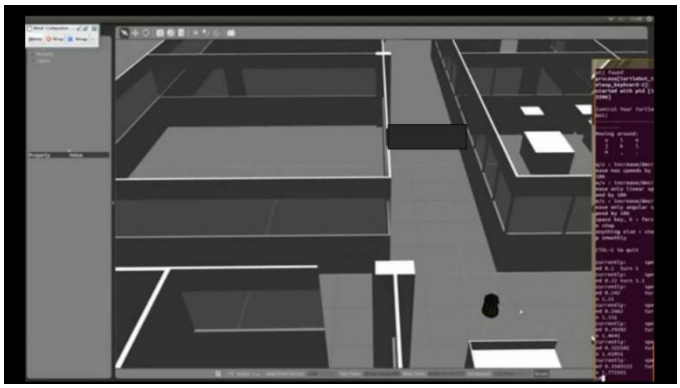


Fig 7: Execution of Agv on the Given Maze

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 03, March-2016

On the same maze an obstacle is introduced because of which AGV takes an alternative path. This is represented by the fig 8

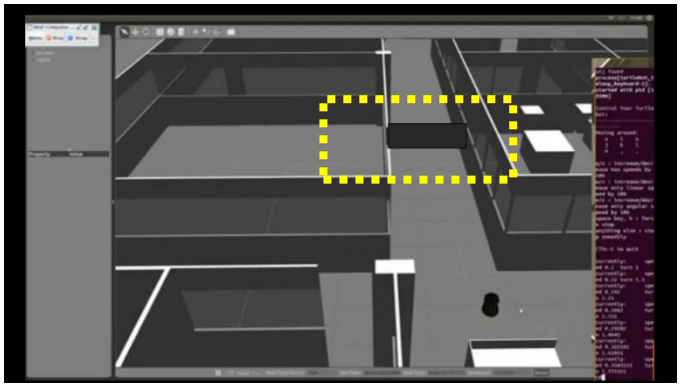


Fig 8 :Takes an alternative path because of an obstacle

As in case of any contradiction between the Dijikstra's algorithm and A* algorithm for retaining the shortest path, according to the proposed design the system follows up the results of A* algorithm as its outcome is more precise. And thus in this scenario the AGV takes the next least distance path by creating a virtual object on its second least distance path during its navigation. This is represented in the fig 9

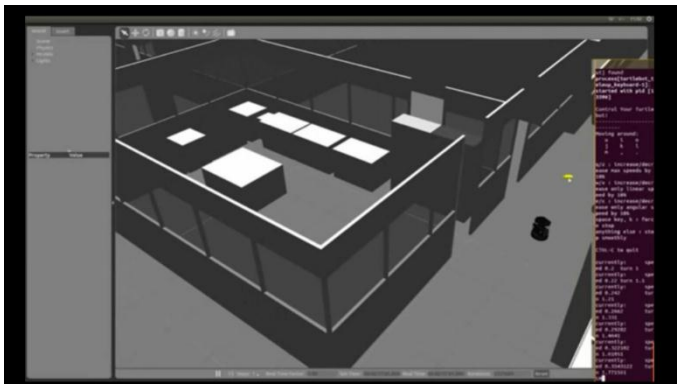


Fig 9: Creation of virtual obstacle in order to take up the shortest path

TABLE 2 COMPARISION TABLE BETWEEN THE EXISTING AND PROPOSED

PERFORMANCE METRIC	AVERAGE VALUE	
	DECENTRALIZED ALGORITHM	MULTI-PATH STORAGE ALGORITHM
SHORTEST PATH	1.5m	1.5m
TRAVELLED PATH	2m	2m
MINIMAL EXECUTION TIME	198.09s	112.57s
TOTAL TIME	220.38s	198.36s

VI. CONCLUSION

This paper proposed Multiple path algorithm that are used for path replanning in case of an obstacle present in the path of AGV. This algorithm serves well in both known and unknown environment. To attain the task, the following were done system modeling, path replanning and controller deign. Its a known fact that as the memory storage is reduced then its corresponding execution time is also reduced. Thus, in this thesis, a detailed description of MULTI-PATH STORAGE algorithm and A* algorithm for the execution of AGV is given. The proposed algorithm renders an efficient way for vehicle's navigation which has been proven by obtaining the time of execution which is 85s faster than the decentralized algorithm. And the navigation is also made wireless which is controlled and configured using the Bluetooth. The obstacle detection is made better by using A* algorithm for selecting the alternative path during the scenario of an obstacle on the navigation path.

The simulation obtained using the Arduino and Gazebo simulator tools the output for the proposed algorithm is obtained. And their output is also obtained in its designed hardware module.

ACKNOWLEDGMENT

I would like to thank Mr. R. Imtiaz Assistant professor ECE department, SMVEC for guiding me throughout the project. He has given me technical support and helped me to finish the project successfully.

REFERENCES

[1] J. Guo, P. Hu, L. Li, and R. Wang, "Design of automatic steering controller for trajectory tracking of unmanned vehicles using genetic algorithms," IEEE Transactions on Vehicular Technology, vol. 61, iss. 7, Sept. 2012.
[2] P.T. Doan, T.T. Nguyen, V.T. Dinh, HK Kim, and S.B. Kim, "Path tracking control of automated guided vehicle using camera sensor," in Proc. of the I" International Symposium on Automotive and Convergence Engineering, pp. 20-26,2011.
[3] D.K. Chwa, "Fuzzy adaptive tracking control of wheeled mobile robots with state-dependent kinematic and dynamic disturbances," IEEE Transactions on Fuzzy Systems, vol. 20, iss. 3,J une 2012.
[4] M. Likachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on suboptimality," Advances in Neural Information Processing System, MIT Press, 2003.
[5] S. Koenig and M. Likachev, "Incremental A*," in Proc. of the Neutral Information Processing Systems, 200 I.
[6] M. Likachev and S. Koenig, "A generalized framework for lifelong planning A* ", in Proc. of the International Conference on Automated Planning and Scheduling, 2005.
[7] D. Ferguson and A. Stentz, "Field D*: An interpolation-based path planner and replanner ", Springer Tracts in Advanced Robotics, vol. 28, pp. 239 - 253, 2007.
[8] S.c. Yun, V. Ganapathy, and T.W. Chien, "Enhanced D* Lite algorithm for mobile robot navigation ", in Proc. of 2010 IEEE Symposium on Industrial Electronics and Applications, 20 10.
[9] S. Koenig and M. Likachev, "Fast replanning for navigation in unknown terrain," IEEE Transactions on Robotics,v ol. 21, no. 3, 2005.
[10] D. Ferguson and A. Stentz, "The delayed D* algorithm for efficient path replanning," in Proc. of the IEEE International Conference on Robotics and Automation, 2005.
[11] S. Koenig, D. Furey, and C. Bauer, "Heuristic search-based replanning," in Proc. of the International Conference on Artificial Intelligence Planning and System, 2002.
[12] T.L. Bui, P.T. Doan, S.S. Park, HK Kim, and S.B. Kim, "AGV trajectory control based on laser sensor navigaiton," International Journal of Science and Engineering, vol. 4, no. 1,p p. 16-20,2013.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)