# Experimental Comparison of A* and D* Lite Path Planning Algorithms for Differential Drive Automated Guided Vehicle

Yuhanes Dedy Setiawan[1], Pandu Sandi Pratama[1], Sang Kwon Jeong[2], Vo Hoang Duy[3], and Sang Bong Kim[1]

[1] Department of Mechanical and Automotive Engineering, Pukyong National University, Yongdang Campus, Nam-Gu, Busan 608-739, Korea
kimsb@pknu.ac.kr
[2] Han Sung WellTech Co., LTD.
#119-5, SamNak-dong, SaSang-Gu, Busan Korea
[3] Department of Electrical and Electronic Engineering, Ton Duc Thang University, Ho Chi Minh City, VietNam

**Abstract.** Nowadays there are some path planning algorithms for mobile robot which have been documented and explained individually in detail such as A*, LPA*, D* and D* Lite. However, there is still a lack of a comparative analysis of these algorithms. Therefore, in this paper a research of comparing A* and D* Lite algorithm for AGV's path planning is conducted by using simulation and experiment. The goal is to compare the characteristic of each algorithm when they are applied in a real differential drive AGV and give the reader a guide in choosing algorithms for their own planning domains. The emphasis of this comparison is on the computation time of generating trajectory and the distance of the generated trajectory. The simulation and experimental results show that generally D* Lite can plan the shorter path with faster computation time than A*. However, there are some cases when D* Lite is less effective than A*. It means which of the algorithms should be chosen depends on the requirement of the system.

**Keywords:** Autonomous Guided Vehicle, Path Planning, D* Lite, A* algorithm.

## 1 Introduction

Automated Guided Vehicle (AGV) is a mobile robot that moves materials around manufacturing facilities or warehouse in industrial application. Thus, AGV has been used widely because it can increase efficiency and reduce cost of material handling. To be able to work autonomously, AGV needs to be able to plan a path for itself. Path planning is a very important step that AGV has to perform. Therefore, there are many researches related to this area.

Path planning is a process of robot planning a path where it should navigate with avoiding obstacles in the surroundings. Robot has to do this process continuously

I. Zelinka et al. (eds.), *AETA 2013: Recent Advances in Electrical Engineering and Related Sciences*, Lecture Notes in Electrical Engineering 282,
DOI: 10.1007/978-3-642-41968-3_55, 

until it reaches the goal position even in the presence of unknown obstacles that suddenly appear in the surroundings. There are a lot of path planning algorithms that have been researched for mobile robot, such as A* [1], LPA* (Lifelong Planning A*) [2], D* [3] and D* Lite algorithms [4].

Due to lack of comparative analysis information of path planning algorithms, choosing the best algorithm to be applied in a particular system is sometimes difficult. Hence the main contribution of this paper is to compare the characteristics of path planning algorithms such that their characteristics can be understood by users. In this research, the characteristics of A* and D* Lite algorithm will be researched and compared. D* Lite algorithm is the most widely used path planning algorithm due to its use of incremental updates and heuristics, whereas A* algorithm is a popular classical graph search algorithm in calculating the least-cost path on a weighted graphs [5].

This paper is organized as follows. In section 2 the path planning algorithms, both A* and D* Lite, are described. In section 3 the system description of experimental model AGV is presented. In section 4 simulation and experimental results are discussed. Conclusion is described in the end of the paper.

## 2 Path Planning Algorithm

In this section, A* and D* Lite algorithms are introduced about how their algorithms work. Firstly, A* algorithm is introduced in Fig. 1. This is a pseudocode of A* algorithm made for user to understand easier. The italic sentence denoted with symbol "//" is brief description of the code. The algorithm is illustrated in Fig. 2(a), which shows an example of simple path planning using A* algorithm. The numbers shown on the node are

**ComputeShortestPath()**

01. while $\left(\arg\min_{s \in OPEN}\left(g(s)+h(s,s_{goal})\right) \neq s_{goal}\right)$ *//check whether* $s_{goal}$ *is reached or not*
02. remove state $s$ from the front of *OPEN*; *//state with the smallest value (becomes path)*
03. for all $s' \in Succ(s)$ *//successor of s to find the other state that has the smallest value*
04. *if* $\left(g(s') > g(s)+c(s,s')\right)$ *//to check whether the s' has been calculated or not*
05. $g(s') = g(s)+c(s,s')$;
06. insert $s'$ into *OPEN* with value $\left(g(s')+h(s',s_{goal})\right)$;

**Main()**

*07.* for all $s \in S$ *//initialization of s as element of S (set of states in finite state space)*
08. $g(s) = \infty$; *//initiate the algorithm by making the g value of all s to be infinity*
09. $g(s_{start}) = 0$; *//make the g value of* $s_{start}$ *= 0 to begin the path finding algorithm*
10. $OPEN = \emptyset$; *//make the OPEN list empty*
11. insert $s_{start}$ into *OPEN* with value $\left(g(s_{start})+h(s_{start},s_{goal})\right)$;
12. ComputeShortestPath();

**Fig. 1.** A* algorithm [5]

explained in Fig. 2(b). g(s) is the sum of the current path cost from the start node, whereas h(s, $s_{goal}$) is heuristic estimate of current path cost to the goal node. The first and second calculations of the algorithm are shown in Fig. 2(c) and 2(d), respectively.

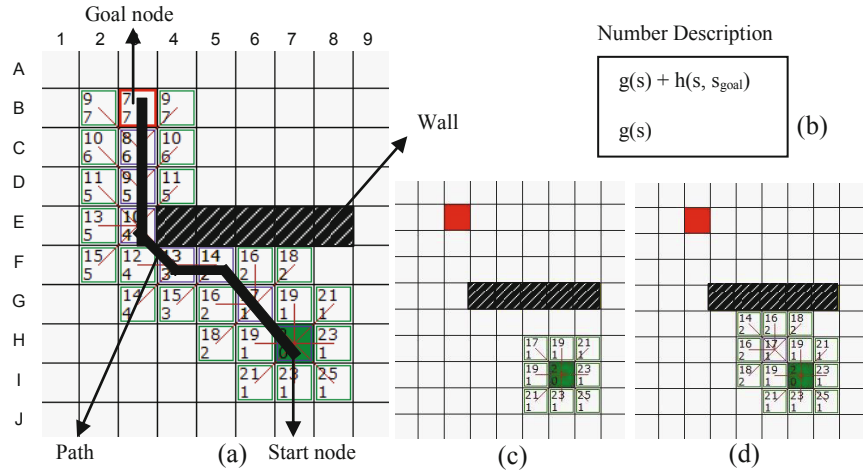


**Fig. 2.** Illustration of A* algorithm path planning

The algorithm begins by calculating the values of the start node as shown in Fig. 2(c). The heuristic in this case is 2 for vertical and horizontal movements and 4 for diagonal movement. Therefore, the numbers in the start node are 2 and 0 to show the heuristic that is used in this case. Then the algorithm will evaluate the neighbors' values of the start node. One of them has values of 17 and 1. Because the goal node is 8 nodes away from that node, the h(s, $s_{goal}$) is 16. Because g value of the node is 1 (calculated in line 5 in Fig. 1), the top number is 17. This node is the start node's neighbor, which has the smallest value of its neighbor values. Therefore, this node will be the path and its neighbors will be evaluated to find the other node that has the smallest value again. If this process will run until the goal node is evaluated, the path planning is finished. Detailed explanation of this algorithm can be found in references [5] for both A* algorithm and D* Lite algorithm path planning.

D* Lite algorithm has a little bit different way in planning the path. The algorithm and illustration of simple D* Lite algorithm path planning are shown in Fig. 3 and Fig. 4, respectively. Basically how D* Lite algorithm works is very similar with how A* algorithm works. Only D* Lite algorithm different from A* algorithm is in the direction of the algorithm work because the D* Lite algorithm is calculated from $s_{goal}$. Furthermore, there is a new parameter, i.e. *rhs* (right hand side) which is a one-step look-ahead cost. Similarly to A*, D* Lite algorithm utilizes a heuristic and a priority queue to perform its search and to order its cost updates efficiently. The heuristic in this algorithm is different with that in the A* algorithm. In D* Lite algorithm, the heuristic value is the distance from the state to $s_{start}$. So if the state is 4 nodes away

from the $s_{start}$, the heuristic value is also 4. Fig. 4(b) shows the number description of the node, whereas Fig. 4(c) and Fig. 4(d) show the first and second calculation of the D* Lite algorithm, respectively. According to Koenig and Likhachev, D* Lite algorithm can be two times more efficient in replanning a new path than A* algorithm because A* algorithm has to replan the path from the beginning whereas D* Lite algorithm do not replan the path because it already has information of the surroundings from the first search [7].

**key(s)**
01. return $[\min(g(s), rhs(s)) + h(s_{start}, s); \min(g(s), rhs(s))]$;

**UpdateState(s)**
02. if $s$ was not visited before
03. $g(s) = \infty$;
04. *if* $(s \neq s_{goal})$ $rhs(s) = \min_{s' \in Succ(s)}(c(s, s') + g(s'))$ ;
05. *if* $(s \in OPEN)$ remove $s$ from *OPEN*;
06. *if* $(g(s) \neq rhs(s))$ insert $s$ into *OPEN* with key($s$);

**ComputeShortestPath()**
07. while$(\min_{s \in OPEN}(key(s)) < key(s_{start})$ OR $rhs(s_{start}) \neq g(s_{start}))$
08. remove state $s$ with the minimum key from *OPEN*;
09. if $(g(s) > rhs(s))$
10. $g(s) = rhs(s)$;
11. for all $s' \in \text{Pred}(s)$ UpdateState$(s')$;
12. else
13. $g(s) = \infty$;
14. for all $s' \in \text{Pred}(s) \cup \{s\}$ UpdateState$(s')$;

**Main()**
15. $g(s_{start}) = rhs(s_{start}) = \infty$; $g(s_{goal}) = \infty$;
16. $rhs(s_{goal}) = 0$; $OPEN = \emptyset$;
17. insert $s_{goal}$ into OPEN with key($s_{goal}$);
18. forever
19. ComputeShortestPath();
20. Wait for changes in edge costs;
21. for all directed edges ($u$, $v$) with changed edge costs
22. Update the edge costs $c(u, v)$;
23. UpdateState($u$);

**Fig. 3.** D* Lite algorithm [5]

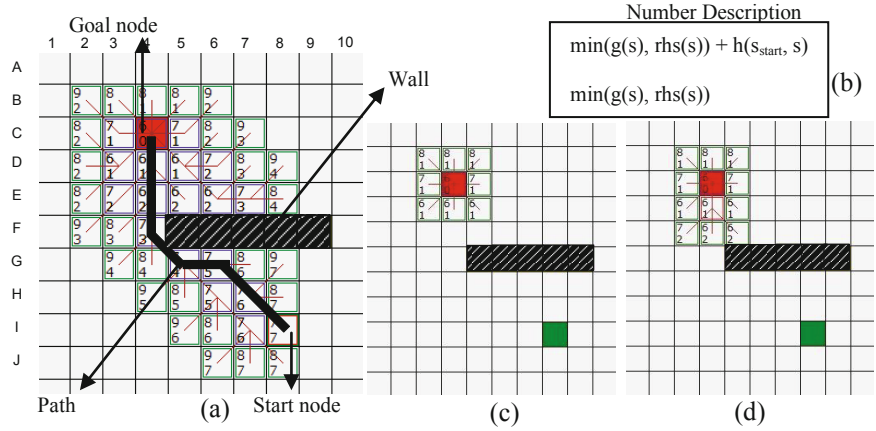


**Fig. 4.** Illustration of D* Lite algorithm path planning

## 3 System Configuration

In this experiment, a differential drive AGV is used as the experimental system. Fig. 5 shows the configuration of the AGV system. The main controller uses Tank-800 industrial computer. The connection for the industrial computer and motor driver uses half PCI MMC Board. The system has four wheels where two wheels are passive castor wheels mounted at the front and back side of AGV and the others are two driving wheels mounted at the left and right side of AGV. The driving wheels are driven by 200W/3,000 rpm BLDC motors. LCD screen used for monitoring process is mounted on the back side of the body. The AGV system is powered by 2 pieces of 12V/8AH batteries.

AGV uses NAV200 for positioning in absolute coordinate and LMS151 for obstacle detection. How the AGV works is explained as follows. First, AGV will make a map by using data from LMS151. From this map, AGV can know the surroundings of the system including the presence of obstacles. Then AGV will measure its position based on data from NAV200 sensor. After that, AGV will execute the path planning algorithm to get the path where AGV should go from current position to goal position. The next step is trajectory tracking of the path that has been built. All of these processes can be monitored from monitor. The controller will send signal to both left and right motor drivers such that they supply voltage signal to the motors to rotate according to the signal from controller. The trajectory tracking of AGV is controlled by feedback control system to make the AGV track the trajectory well.

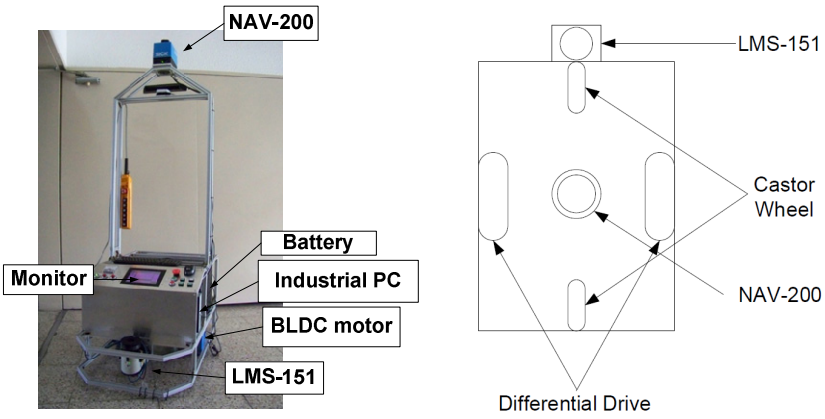


**Fig. 5.** Configuration of AGV System [8]

If there are unknown obstacles that suddenly appear in the surrounding, the LMS 151 will detect the object and path planning algorithm will plan a new path. This process is called path replanning and performed continuously until the AGV reaches the goal position safely.

## 4 Simulation and Experimental Results

The experimental comparison was conducted in two ways, i.e. simulation and experiment. In the simulation, comparison was conducted by comparing the computation time needed for generating trajectory and the distance of the generated trajectory that A* and D* Lite algorithms make. After the path is planned by both A* and D* Lite algorithm, the experimental comparison is conducted by applying the generated path to the AGV.

Fig. 6 shows the simulation results generated by A* and D* Lite path planning algorithms. The result of A* path planning algorithm is shown in Fig. 6(a), whereas the result of D* Lite algorithm in Fig. 6(b). How the path is built is explained in section 2. As it can be seen that D* Lite algorithm generates the shorter distance path from start point to goal point with faster time than A* algorithm. A* algorithm generates the path with 20 nodes in 0.0048 sec, whereas D* Lite algorithm generates the path with 17 nodes in 0.0028 sec. It can be concluded that D* Lite algorithm is able to plan a shorter path with faster time than A* algorithm.

The next comparison is conducted by experiment. Fig. 7 shows simple experimental results that make the AGV to plan a path from start node to goal node by avoiding an obstacle using A* algorithm (a) and D* Lite algorithm (b). But this experiment can show that D* Lite algorithm can also be less effective than A*

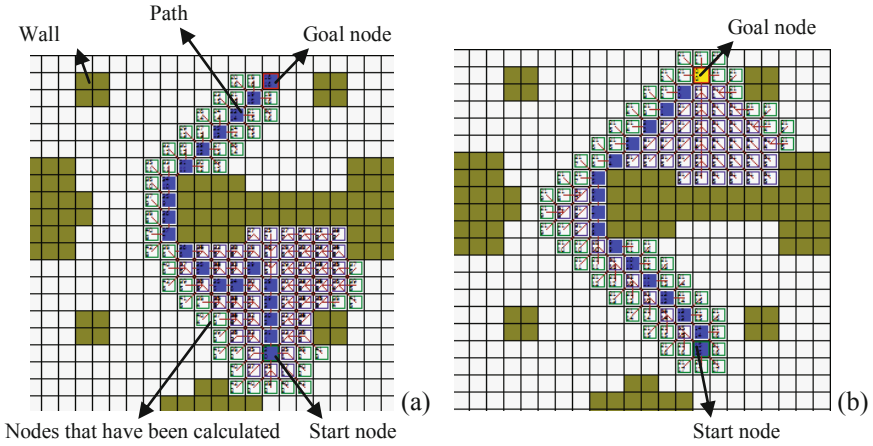


**Fig. 6.** Simulation results of A* algorithm (a) and D* Lite algorithm (b)

algorithm because D* Lite algorithm calculates more nodes to plan the shortest path with the purpose of replanning the path in case that unknown obstacles suddenly appear. Fig. 8(a) shows that the AGV trying to avoid obstacle, and Fig. 8(b) shows that it can successfully avoid the obstacle along to the planned path in experiment.

In this case, both A* algorithm and D* Lite algorithm generate the path with 15 nodes from start point to goal point. But A* algorithm generates the path only in 0.0003 second, whereas D* Lite algorithm in 0.0052 second. However, if there is unknown obstacles in the path that make the algorithm to replan, D* Lite algorithm will be more effective because it only needs to replan with data that have been calculated before, whereas A* algorithm has to replan and calculate the path from the beginning that makes the replanning process longer than D* Lite algorithm. An example of path replanning using A* algorithm and D* Lite algorithm are shown in Fig. 9 and 10, respectively.

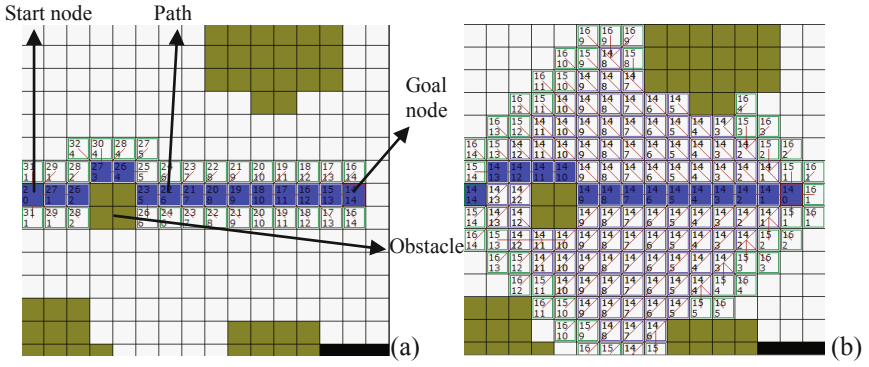


**Fig. 7.** Experimental results of A* algorithm (a) and D* Lite algorithm (b)

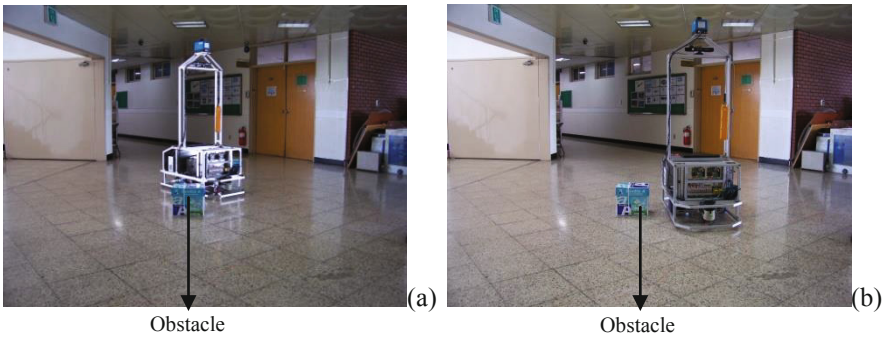


**Fig. 8.** Experiment results of path planning algorithm using differential drive AGV

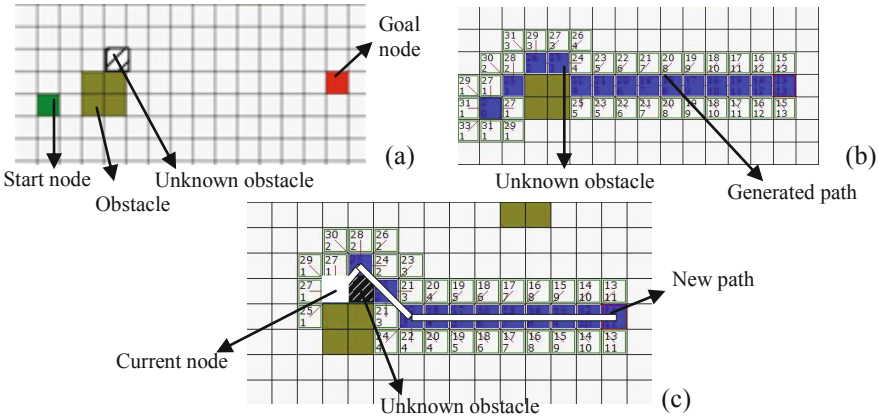


**Fig. 9.** Simulation results of A* path replanning algorithm

Basically, A* and D* Lite path replanning algorithms work similarly. The two replanning algorithm processes are compared in this section. Firstly, the start node of the robot is defined as shown in Fig. 9(a) for A* algorithm and Fig. 10(a) for D* Lite algorithm. Because at the start state the robot does not know that there is unknown obstacle in the surroundings, the robot plans the shortest path to the goal state that passes the unknown obstacle, which is illustrated in both Fig. 9(b) and Fig. 10(b). After the path is built, the robot tracks the generated path until the robot meets the unknown obstacle, as illustrated in Fig. 9(c) and Fig. 10(c). Because the robot cannot pass through the obstacle, it replans the path by using information data that have been obtained in the first search for D* Lite algorithm, whereas A* algorithm will replan the path from the beginning. Finally, the robot replans a path illustrated in Fig. 9(c) and Fig. 10(d) and tracks the new path to the goal state.

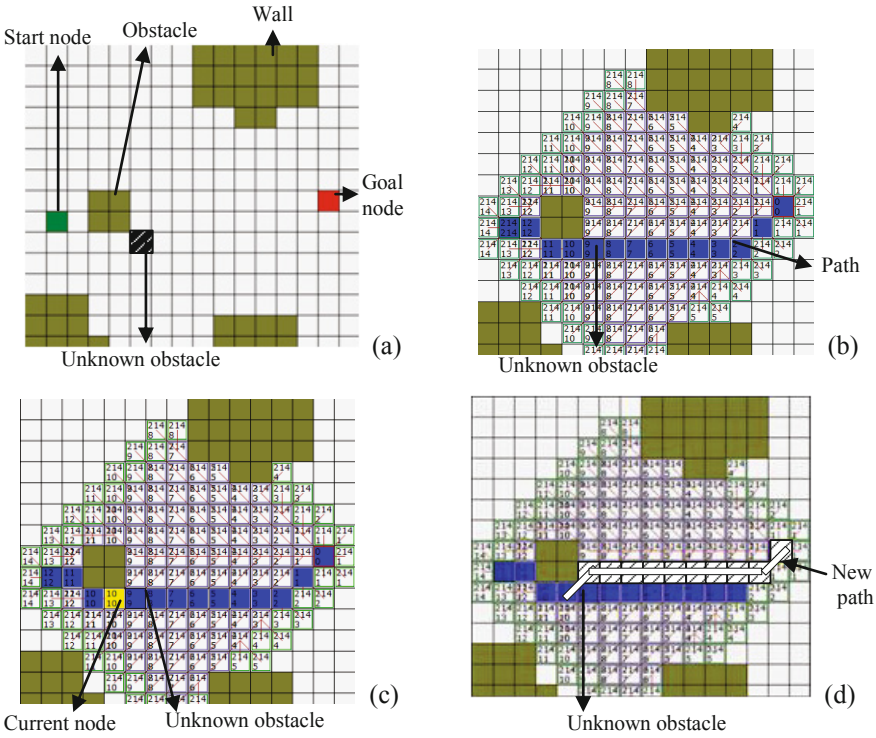


**Fig. 10.** Simulation results of D* Lite path replanning algorithm

In replanning case, which of algorithms is better depends on the size of the state space. If the state space has a big size, D* Lite algorithm is better than A* algorithm by giving a faster path replanning. This is because D* Lite algorithm evaluates more nodes at the first search such that replanning the path can be very fast, whereas A* algorithm has to replan it from beginning. However if the state space is small and simple like the example of replanning in Fig. 9 and 10, the A* algorithm will be more effective than D* Lite algorithm because D* Lite algorithm evaluates too many nodes. In that example, the total time is needed for A* algorithm to replan is 0.0047 second, whereas D* Lite algorithm takes 0.2988 second.

## 5 Conclusion

This paper presented simulation and experimental comparison results of A* algorithm and D* Lite algorithm for path planning of AGV. This research was done to give a guide for users in choosing which path planning algorithm should be used in a certain system. The simulation and experimental results show that D* Lite algorithm can plan the shorter path in a shorter time than A* algorithm because D* Lite algorithm uses heuristic to restrict the replanning process to only states that are relevant for repairing

the path. However, D* Lite algorithm can also be less effective than A* algorithm in some cases. Thus the use of these algorithms depends on the requirement of a system.

**Acknowledgments.** This research was supported by a grant from Construction Technology Innovation Program (CTIP) funded by Ministry of Land, Infrastructure and Transport (MOLIT) of Korean Government.

## References

1. Nilsson, N.: Problem-Solving Methods in Artificial Intelligence. McGraw-Hill (1971)
2. Koenig, S., Likhachev, M.: Incremental A*. In: Proceedings of the Neutral Information Processing Systems (2001)
3. Stentz, A.: The Focused D* Algorithm for Real-Time Replanning. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1652–1659 (1995)
4. Koenig, S., Likhachev, M.: D* Lite. American Association for Artificial Intelligence (2002)
5. Ferguson, D., Likhachev, M., Stentz, A.: A Guide to Heuristic-based Path Planning. American Association for Artificial Intelligence (2005)
6. Koenig, S., Likhachev, M.: Fast Replanning for Navigation in Unknown Terrain. IEEE Transactions on Robotics 21(3) (2005)
7. Koenig, S., Likhachev, M.: Improved Fast Replanning for Robot Navigation in Unknown Terrain. In: Proceedings of the IEEE International Conference on Robotics and Automation-ICRA (2002)
8. Pratama, P.S., Jeong, S.K., Park, S.S., Kim, S.B.: Moving Object Tracking and Avoidance Algorithm for Differential Driving AGV Based on Laser Measurement Technology. International Journal of Science and Engineering 4(1), 11–15 (2013)