Collision Avoidance Algorithm of Mobile Robots at Grid Map Intersection Point

Young-In Choi¹ , Jae-Hoon Cho² , and Yong-Tae Kim³

¹Nova Co. Ltd., Anseong, Korea

²Smart Logistics Technology Institute, Hankyong National University, Anseong, Korea

³Department of ICT, Robot and Mechanical Engineering, Hankyong National University, Korea

ljfis

Abstract

In this study, we propose a collision avoidance algorithm for mobile robots that can be applied to distribution centers. The proposed algorithm effectively avoids the collision of several robots, moving simultaneously in a distribution center. It can be applied where the driving environment of the robots is composed of a grid map, and the path planning of individual robots is generated by the D* Lite algorithm. The proposed algorithm operates in a way that the new path and movement strategy of individual robots are reset when the collision of multiple robots is expected at any grid point. It operates on a central server and transmits the paths of individual robots. Simulation was performed in this study to evaluate the performance of the proposed algorithm, and the results showed that collision was effectively avoided in each situation.

Keywords: Collision avoidance algorithm, Distribution center, Path planning, D* Lite algorithm

1. Introduction

Currently, various path planning algorithms are being used in the logistics industry. Path planning includes global and local path planning. Global path planning uses the given information to avoid collisions with fixed obstacles, and reaches the desired destination with minimal cost. On the other hand, local path planning avoids obstacles by modifying the route in real time [1]. In general, local path planning has a disadvantage of falling into the local minimum; therefore, it is necessary to study an algorithm that has been integrated with the global route planning. However, the fused path planning algorithm requires a lot of computation for various collision situations, such as the frontal or side collision situation shown in Figure 1, where two robots collide at an intersection point; collision may also be with another robot in the process of avoiding a fixed obstacle [2]. When selecting the initial driving route, conventional path planning algorithms only consider the minimum cost. Therefore, while driving, unnecessary costs may be incurred when avoiding obstacles [3–5].

To solve this problem, potential collisions as well as costs should be considered when selecting an initial driving path. In this study, the whole path is composed of a grid-map, and the initial movement path of the individual robots is selected by the D* Lite algorithm. Additionally, it is possible to re-select the path movement strategy after collision detection at the grid point. We propose a collision detection algorithm that can reduce the amount of computation in the central server with minimal detection.

Received: May 13, 2020

Revised : Jun. 4, 2020

Accepted: Jun. 6, 2020

Correspondence to: Yong-Tae Kim

(ytkim@hknu.ac.kr)

©The Korean Institute of Intelligent Systems

©This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Figure 1. Cross point collision situation.

2. Path Planning Using D* Lite Algorithm

2.1 Path Re-planning between Two Robots

Selecting the initial path of the robot using only the classic D* Lite algorithm can cause some problems in resetting the path to avoid obstacles. First, the D* Lite algorithm recognizes other robots moving along the paths as obstacles and creates a bypassing pass, while other robots approaching from the other side also create a bypassing pass. This situation incurs unnecessary driving costs for two robots that do not share an intersection.

In Figure 2, Robot 1 recognizes Robot 2 as an obstacle and resets the path to avoid the obstacle. Additionally, Robot 2 recognizes Robot 1 as an obstacle and selects a new path. Therefore, the travelling path is increased by at least two steps from the initial path for both robots.

Figure 2. Re-planning of bypassing pass.

Figure 3. Collision situation of robots along the same path.

Figure 4. Side impact situation.

2.2 Collision on the Same Path

When two robots are recognized as obstacles as shown in the aforementioned case, the generation of additional movement paths presents a problem. However, as shown in Figure 3, a repetitive path reset occurs as the robots move on the initial path with the intersection. Whenever the robots move past one grid point, nearby robots are recognized as obstacles, causing unnecessary path resets.

2.3 Side Collision Situation

It is known that the D* Lite algorithm can detect and avoid dynamically moving obstacles. However, the dynamic obstacle can be avoided only when it is on the driving path of the robot; however, a collision may occur if a robot is approaching from the side (Figure 4).

Figure 5. Structure of the proposed algorithm.

3. Collision Detection and Path Planning

3.1 Basic Structure of the Proposed Algorithm

The aim of the general collision avoidance algorithm is to recognize a moving robot on the movement path as an obstacle and avoid it correctly. However, general collision avoidance algorithms do not respond effectively to the situations mentioned in Section 2.

Therefore, a collision prediction system is needed to effectively respond to these situations, and reduce the amount of computation needed to reset the driving path.

The proposed algorithm receives real-time information from the robot and then predicts the collision points through a minimum area search (Figure 5). The range of the minimum area search is determined by the designer and can be set from 5 to 10 grid points. If a collision point exists in the path of the robots, the collision situation is determined, i.e., collision along the same path (frontal collision) or collision from the side (side collision). The collision situation determination algorithm of the proposed algorithm is functionally divided into an emergency collision prediction situation, and a general collision prediction situation, and determines whether it is a frontal or side collision, respectively. When the type of collision is determined by the path planning algorithm operated in the main server, the paths are reset for each situation and transmitted to the robots.

3.2 Setting Priorities among Robots

When the paths of the robots share an intersection where collision is predicted, computation and consumption costs may be reduced if the path of one of the robots is not reset. As shown in the situation in Figure 2, the time loss can be reduced if the two robots do not attempt to reset the path simultaneously, and only one of them resets the path to avoid the other robot. Therefore, in the proposed algorithm, it is necessary to decide which of the robots needs to reset their path before collision.

In the proposed algorithm, the robots re-planning the route are determined based on priority (which can be assigned by a human operator based on the moving distance, work time, and job importance). However, for the effective operation of all robots, the maximum number of path resets must be limited. When the number of times for resetting a path is greater than or equal to a value determined by the human operator, the priority is set to 0 as the highest priority. For example, if the human operator sets the number of times to 5, the robot has the highest priority if it has modified the path 5 times to avoid other robots on the path from the starting point to the target point. The proposed algorithm determines the priority using these conditions and solves the problems caused by all bypassing robots when the collision situation between them is predicted.

3.3 Minimum Area Search

While moving in the grid map environment, the robot scans the QR code on the floor and transmits information, such as current heading direction, the scanned QR code, and the next moving direction, to the management server. The management server stores the received information in a separate memory space and uses it to predict the collision situation. The collision situation prediction method compares the next path of the highest priority robot with the next path of other robots. Because the proposed algorithm searches the grid points in only four directions from the current position of the robot, it can reduce the amount of computation needed to search for collisions as shown in Figure 6.

The method for predicting the frontal and side collision situations is designed using information about the current position of the robots transmitted to the server and the surrounding grid points. In the collision prediction situations, the locations of the lattice points in the four directions are identified in advance, and the management server conducts calculations to determine robots having the same values.

Emergency collision occurs when the next coordinate of the

Figure 6. Minimum area navigation.

Figure 7. Emergency collision forecasting situation.

Figure 8. Safe area checking.

high priority robot is the same as the current coordinate of another robot, as shown in Figure 7. In the case of a frontal collision under an emergency collision prediction situation, the low priority robot resets the route, avoiding the high-priority robot. In the side collision prediction situation, the robot having the current grid point of the robot matching the next grid point is avoided by moving one step, regardless of the priority.

3.4 Safety Zone Verification

The safety zone verification procedure verifies the safety of a region where a low priority robot can avoid a high priority robot when the collision situation is predicted by searching for the minimum area. Figure 8 shows a front collision situation by two robots. If Robot 1 is a low-priority robot, the safety area verification procedure first checks the safety in all possible directions.

The safe area has priority over the area where the robot can move immediately. In a frontal collision situation, the safety zone first checks the left or right side, because the robots move

Figure 9. Safe zones in various situations.

directly to obstacles. If there are obstacles on both the left and right sides of the robot, the robot moves backwards and checks the left and right sides again.

3.5 Safety Zone Verification according to the Situation

The safety zone verification procedure depends on the collision prediction situation. Figure 9 shows the safe areas that can be selected depending on the situation. In Figure 9, Sol1 and Sol2 are candidates for the safety zone, respectively. When robots have a frontal collision, the safety zone is identified based on the robot with the lowest priority. In the side collision, the robot with the lower priority waits as the higher priority robot checks the safety zone. Moreover, in the side collision, positions are searched two steps forward, as well as on the left and right positions of the current robot. If there are no obstacles when checking two steps forward, the robot with the highest priority moves to the corresponding location immediately.

3.6 Resetting the Driving Path

When the robot moves to a safe area, it deviates from the initial path. Therefore, the robot requests a path reset from the management server based on the current location. The robot then continues movement using the newly set path. Figure 10 shows the process of resetting the path of a low-priority robot, and movement when a collision situation is predicted. Unlike the D* Lite algorithm, the proposed algorithm resets the path after evading movement to the safe area when the robot resets the path.

For the high priority robot to move quickly without loss of movement, the low priority robot waits while the high priority robot moves. The proposed algorithm resets the path when an

Figure 10. Front collision path reset process.

Figure 11. Reset process of front collision path.

obstacle exists in the path of the high priority robot, as shown in Figure 11.

4. Simulation and Results

In this simulation, the grid map is generated within a 10×10 range, and a collision prediction situation is created. The collision prediction situation includes a frontal collision, a side collision, and collision in a safe area when evading.

Figures 12-14 show detailed flow charts of each collision situation. The maximum number of robot collisions in one intersection is four. When two robots collide, frontal and side collision situations may occur. The processing algorithm in this case is shown in Figure 12. In the case of a frontal collision situation, the robot with the highest priority waits, and the non-priority robot checks the right and left sides for avoidance. The non-priority robot checks the right side and moves there when safety is confirmed. If there is an obstacle on the right side, the robot is configured to check the left side and moves there when

safety is confirmed. If there are obstacles on both the right and left sides, the robot goes back and resets the path. The collision situation with three robots is determined by the complex and side collision situations based on the position of the priority robot. Complex collision considers both the frontal and side collisions, whereas the side collision considers only the left and right side collisions.

In the three collisions, the collision robots wait first, and in the case of a complex collision, the frontal collision is processed first, followed by the side collision. Figure 13 shows the avoidance flowchart in the event of a collision with three robots. Basically, it is the same as the method used for the two robot collisions; however, after resolving all collisions, the verification process is required once more. The path is then reset when it is determined that all the robots are safe from the collision situation.

The collision situation of four robots is a complex collision situation for all the robots. This collision situation is also resolved beginning with the frontal collision. Figure 14 shows the avoidance algorithm in the case of a collision of four robots. The robot with the highest priority avoids the frontal collision situation first, and then resolves the frontal and side collision situations in order of priority.

4.1 Simulation on Frontal Collision Avoidance

In the case of a frontal collision, the low priority robot is evacuated to a safe area after collision is detected. Additionally, in the simulation of the side collision situation, the lower priority robot waits as the higher priority robot moves to the safe area. If there is an obstacle when the robot moves to the safe area, it is detected and another safe area is identified. The robot then moves to the safe area and is tested for resetting (Figure 15).

In the results of the frontal collision avoidance simulation, the lower priority robot confirmed the safe area, then moved for avoidance. It was confirmed that the robot received the reset path from the management server and operated correctly. Further, it was confirmed that the high priority robot moved efficiently according to the set path.

4.2 Simulation on Side Collision Avoidance

The results from the side collision avoidance simulation, confirm that the higher priority robot moves after verifying the safe area while the robot with the lower priority waits. Additionally, it is confirmed that the low priority robot moves safely after the

Figure 12. Avoidance algorithm in the event of a collision with two robots.

Figure 13. Avoidance algorithm in the event of a collision with three robots.

high priority robot safely passes the collision prediction point (Figure 16).

5. Results of Frontal Collision Avoidance

To compare the performance of the D* Lite algorithm and the proposed algorithm, simulation was conducted by dividing the distance from the starting point to the target point into 4 steps, 6 steps, and 8 steps. For the convenience of the simulation, the

robot with the lower ID number was used as the high-priority robot. As shown in Table 1, it is confirmed that all the high-priority robots have a 2-step difference in the moving distance. However, the low priority robot that should be avoided is similar to that of the D* Lite algorithm.

At different distances, the difference from the D* Lite algorithm is the same. In the case of a frontal collision, it can be seen that the distance and time of movement of a higher priority robot is saved regardless of the distance.

Figure 14. Avoidance algorithm in the event of a collision with four robots.

Figure 15. Results of frontal impact test. (a) (b) (c) (d).

Table 1. Characteristics Comparison of distance results for the front collision simulation

Step	ID	Distance	D* Lite	Proposal	Comparison
Case 1	R1	4	6	4	-2
	R2	4	6	6	0
Case 2	R1	6	8	6	-2
	R2	6	8	8	0
Case 3	R1	8	10	8	-2
	R2	8	10	10	0

5.1 Results of Side Collision Avoidance

The criteria for the distance traveled in the side collision simulation are similar to those of the frontal collision simulation. In the side collision test results, the D* Lite algorithm recognized only obstacles in the path; therefore, it is disadvantageous because it could not identify dynamic obstacles as obstacles.

Figure 16. Results of side collision experiment. (a) (b) (c) (d).

Consequently, collision occurred regardless of the distance (Table 2).

On the other hand, the proposed algorithm avoids collisions because it checks the position of each robot, and predicts the collision situation. In the case of a frontal collision, there was only a situation where the lower priority robot waited without a separate avoidance situation; thus, the moving distance was the same as the given distance. With regards to time, because there is a waiting time unlike the moving distance, it was observed that the lower priority robot delayed more.

From the results of the side collision simulation, it can be observed that the proposed algorithm is superior to the D* Lite

Table 2. Comparison of travel time results for frontal collision simulation

	ID	Distance	D* Lite	Proposal	Comparison
Case 1	R1	4	30	18	-12
	R2	4	30	30	0
Case 2	R1	6	38	26	-12
	R2	6	38	38	0
Case 3	R1	8	46	34	-12
	R2	8	46	46	0

algorithm in terms of the stability of the collision.

6. Conclusion

In this study, we proposed a method to safely reset the optimal route of a moving robot, after avoiding collision on encountering a collision situation at an intersection. The proposed algorithm involved searching the safety area to effectively avoid collision while the robot moved from the starting point to the target point and ensuring an appropriate response in various collision situations.

To compare the performance between the D* Lite algorithm and the proposed algorithm, simulation was conducted by dividing the distance from the starting point to the target point into 4 steps, 6 steps, and 8 steps. From the simulation results, the superior performance of the proposed algorithm to that of the existing D* Lite algorithm was demonstrated.

Conflict of Interest

No potential conflict of interest relevant to this article was reported.

Acknowledgements

This work was supported by the Gyeonggi-do Regional Research Center program of Gyeonggi Province (No. GRRC-Hankyong 2017-B01, Research Center for Smart Logistics Technology).

References

- [1] Y. H. Jung, H. W. Park, S. J. Lee, and M. C. Won, "Development of a navigation control algorithm for mobile robots using D* search and fuzzy algorithm," *Transactions of the Korean Society of Mechanical Engineers A*, vol. 34,

no. 8, pp. 971-980, 2010. <https://doi.org/10.3795/KSME-A.2010.34.8.971>

- [2] K. Jung, J. Kim, T. Jeon, S. Kim, and K. Kim, "Collision avoidance of multiple path-planning using fuzzy inference system," in *Proceedings of KIIS Spring Conference*, vol. 19, no. 1, pp. 11-14, 2009.
- [3] D. H. Lee, H. J. Kwak, J. H. Kim, C. K. Kim, and G. T. Park, "Optimal path planning algorithm for mobile robots using local path modification," in *Proceedings of CICS 2012 Information and Control Symposium*, Anseong, Korea, 2012, pp. 67-68.
- [4] S. Koenig and M. Likhachev, "D* Lite," in *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, Edmonton, Canada, 2002, pp. 476-483.
- [5] E. H. Jung, J. H. Cho, and Y. T. Kim, "Collision avoidance algorithm using priority of logistics transportation robot," in *Proceedings of KIIS Spring Conference*, vol. 28, no.1, pp. 61-62, 2018.

Young-In Choi received a master's degree. In 2003, he received an associate degree from Yongin Songdam College. In 2008, he received a bachelor's degree from the credit banking system and in 2020, a master's degree from Hankyung National University. He worked in the field of RF systems, humanoid development and automation processes from 2007 to 2017. Areas of interest include path analysis algorithms, automated processes, industrial robots, and application programs. E-mail: forv79@gmail.com

Jae-Hoon Cho received his B.S. and M.S. degrees in Control and Instrumentation Engineering from Hanbat National University, Daejeon, Korea, and Ph.D. degree in Control and Instrumentation Engineering from Chungbuk National University, Cheongju, Korea, in 2002, 2005, and 2011, respectively. He has joined in 2011 as a research professor of Smart Logistics Technology Research Center. His current research interests include intelligent robots, logistics automation, renewable energy and its applications. E-mail: jhcho@hknu.ac.kr

Yong-Tae Kim received his B.S. degree in Electronic Engineering from Yonsei University, Seoul, Korea, and his M.S. and Ph.D. degrees in Electrical and Electronic Engineering from KAIST, Daejeon, Korea, in 1991, 1993, and 1998, respectively. From 1998 to 2000, he was a senior engineering in Samsung. In 2006, he was a visiting scholar at Department

of Electrical Engineering in University of Illinois at Urbana-Champaign. Since 2002, he has joined the faculties of the Department of Electrical, Electronic and Control Engineering at Hankyong National University, where he is currently a professor. His current research interests include intelligent robots, logistics automation, intelligent control and its applications.

E-mail: ytkim@hknu.ac.kr