Collision Avoidance Method for Mobile Robot Considering Motion and Personal Spaces of Evacuees

Takeshi OHKI, Keiji NAGATANI and Kazuya YOSHIDA

***Abstract*— In the case of disasters such as earthquakes or Nuclear/Biological/Chemical(NBC) terrorist attacks, mobile robots, called "rescue robots," that can work in dangerous environments instead of rescue crews in rescue missions, can be of great help. However, realizing such robot systems requires many types of technologies. In particular, path planning is an important technology that provides a mobile robot with autonomous navigation to a target destination with collision avoidance. To avoid evacuees, the robot should consider the motion of people in the near future. In this research, we propose a collision avoidance method that estimates the motions and personal spaces of the evacuees. The method consists of three steps: "estimation," "conversion," and "planning." In the estimation step, the future positions of evacuees are estimated by considering their planned motions and personal spaces. Then, in the conversion step, a time axis is added to construct a 3D time-space coordinate system. Finally, in the planning step, a distance-time transform is applied to plan a safe 3D path from the robot's current position to the desired goal. The proposed method has been implemented on our rescue robot simulator, and some simulation experiments were conducted to verify its usefulness.**

I. INTRODUCTION

A. Background

In the case of disasters, such as earthquakes or NBC terrorist attacks, it is very dangerous for rescue crews to search for victims at disaster sites that have the potential for secondary disasters. In such a situation, remote controlled mobile robots, called "rescue robots" can be of great help when searching inside collapsed buildings, taking the place of the rescue crews. On the basis of such social demands, research and development activities on rescue robots have increased all over the world. Our research group has also been developing rescue robots, called "Kenaf", to explore underground malls in an NBC terrorist attack scenario [1]. To undertake searching missions in the above scenario, we are currently researching on teleoperation methods, communications, locomotion methods on rough terrains, 3D mapping technologies, and autonomous navigation methods, using this robots as a research platform. One of the most important technologies involves autonomous navigation because skilled operators of teleoperated mobile robots may be in short supply in the case of a wide area disaster. In the case of NBC terrorist attacks, a robot may come across evacuees when it explores the target environment. In such a case, the robot should have a function that allows it to autonomously avoid moving obstacles(evacuees) and prevent collision. It is called the "collision avoidance problem in dynamic environments".

Manuscript received March 10, 2010. This work was supported in part by New Energy and Industrial Technology Development Organization(NEDO).

T. Ohki is with the Department of Aerospace Engineering, Graduate School of Engineering, Tohoku University, Aramaki aza Aoba 6-6-01, Sendai, 980-8579, JAPAN takeshi@astro.mech.tohoku.ac.jp

K. Nagatani is with the Faculty of Aerospace Engineering, Department of Aerospace Engineering, Graduate School of Engineering, Tohoku University, keiji@ieee.org

K. Yoshida is with the Faculty of Aerospace Engineering, Department of Aerospace Engineering, Graduate School of Engineering, Tohoku University, yoshida@astro.mech.tohoku.ac.jp

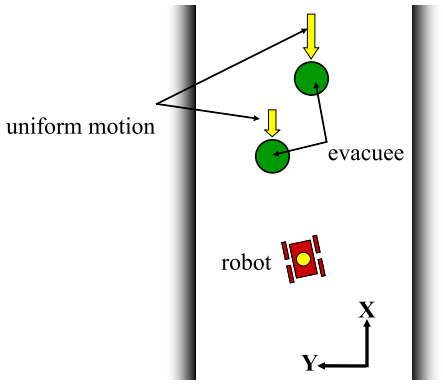


Fig. 1. Flat underground walkway environment

B. Research Purpose

One of the basic methods of solving the above collision avoidance problem is to repeat path planning in a static environment based on sensing data in short cycles. This is a very simple and effective method. However, it is obvious that this strategy generates redundant and unnecessary paths for a mobile robot, and in the worst case, it collides with moving obstacles (we reenacted in our mobile robot simulator). Therefore, it is important to foresee the motion of obstacles in the near future to plan the robot's path. Therefore, in this research, we aim to propose a collision avoidance method that estimates the motion of moving obstacles.

To estimate the motion of moving obstacles, it is very important to determine the type of moving obstacle. Many researches on the collision avoidance problem in dynamic environments assume that the orientation and velocity of obstacles remain constant. However, in the case of evacuees in a disaster environment, this assumption is not sufficient because they also plan their paths to prevent colliding with each other. Therefore, to proceed with our path planning research, we selected specific target scenario of an NBC terrorist attack in an underground mall. The target environment is a flat underground walkway (Fig. 1). In such an environment, we assume that a rescue robot moves from one end of the walkway to the other, and evacuees will move

along the walkway in the opposite direction. Therefore, it is necessary for the robot to avoid evacuees.

In order to realize the above objective, we propose a collision avoidance method that estimates the motions and personal spaces of the evacuees. This method consists of three steps: "estimation," "conversion," and "planning". In the estimation step, it estimates the future positions of evacuees by considering their planned motions and personal spaces. Then, in the conversion step, a time axis is added to construct a 3D time-space coordinate system. Therefore, the 2D obstacle avoidance problem is converted to a 3D path planning problem. Finally, in the planning step, a distance-time transform is applied to plan a safe 3D path from the robot's current position to the desired goal.

To evaluate the validity of the above method, in this research, we implemented it on our rescue robot simulator. In this paper, we would like to introduce our collision avoidance method, and show some simulation results using the rescue robot simulator.

C. Related Works

A large number of studies have been conducted on collision avoidance for mobile robots. Therefore, we introduce some works that are closely related to our study.

Tsubouchi, et al. [2] defined a "3D XYT space" to represent 2D obstacle motions with a time axis for motion planning to avoid collisions in dynamic environment. They assumed all obstacles' motions to be uniformed linear motions. Thus, polygonal obstacles were represented by tilted polygonal prisms. They verified the validity of their method by a simple simulation.

Gupta and Jarvis [3] also used a 3D XYT space, and their planning method used a distance transform [4]. To estimate the motions of moving obstacles, they used a Gaussian distribution to handle the uncertainty of the obstacles' motions. They verified a validity of their method by a simple simulation.

Basically, in this research, we use a 3D XYT space to represent the obstacles' motions, the same as the above two ideas. A feature of our method is to estimate the obstacles' motions by the motion planning of each obstacle considering their personal space, instead of linear or Gaussian assumption.

The concept of the personal space is widely used in studies in the field of psychology, civil engineering and construction, and robotics (e.g. [5],[6]). It is a useful model for estimating the motion of people, and some researches have verified its validity [7]. Each pedestrian has a single personal space, as the extended ego. In this model, in a case where a personal space is invaded, the individual feels uncomfortable and changes his motion to maintain his personal space, as shown in Fig. 2.

II. Overview of Collision Avoidance Method

Firstly, we introduce an overview of our novel collision avoidance method to avoid evacuees in a 2D walkway

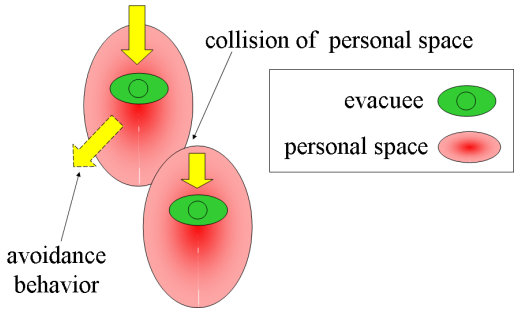


Fig. 2. Collision of personal spaces

environment. The following assumptions are made in this research.

1) The target environment is a flat underground walkway.
2) A robot moves from one end of the walkway to the other, and evacuees move in the opposite direction.
3) The robot can obtain accurate motion vectors for obstacles using its onboard sensors.
4) The robot can follow the planned path accurately.

Based on the above assumptions, in this research, we propose a collision avoidance method that consists of three steps: estimation, conversion, and planning. By periodically repeating the above three steps and executing the planned motion, the robot performs collision avoidance motion.

A. Estimation

The first step "estimation" calculates the future position of each obstacle from its past motion. The features of this method are that (1) it considers not only the movement vectors of the obstacles but their planned motion to avoid colliding with other obstacles, and (2) it considers the concept of personal space of each obstacle. Details of this step are explained in section III.

B. Conversion

The second step conversion is to construct the a 3D time-space coordinate system to express the obstacles' motion from the 2D planar space by adding a time axis T. Therefore, the 2D obstacle avoidance problem is converted into a 3D path planning problem, as shown in section IV.

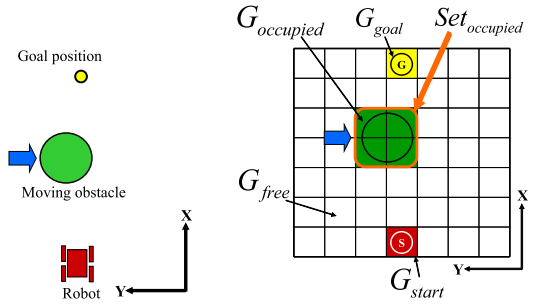
C. 3D Planner

In the third step, planning, a 3D path is planned in the 3D time-space coordinate system. To plan the path, we apply the distance-time transform method. The planned path is equivalent to the planned robot's motion, including its position and time. The details of this step are explained in section V.

III. Estimation

In the first step, estimation, our method estimates the future positions of moving obstacles while considering their personal spaces.

A personal space is generally defined as an oval figure, with the space in front of a person larger than the rear



(a) 2D environment (b) Grid layer

Fig. 3. Definition of grid layer

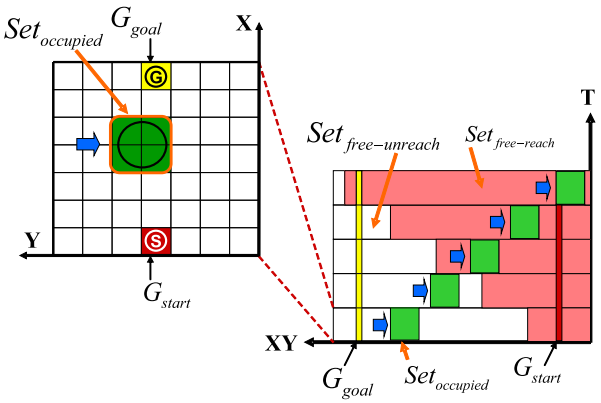


Fig. 4. Definition of grid stack

space [6]. The personal space concept is based on many complex rules, and its shape and size are affected by several factors such as gender, age, and social position. However, to simplify our method, we set the personal space as a constant rectangular in shape, and the front length, rear length and width are 1.5 (m), 0.4 (m), and 0.4 (m), respectively.

In our assumption, all the obstacles move in the same direction at different speeds. Therefore, a slow obstacle in the front should be passed by a fast obstacle in the rear. In this case, in our method, the fast obstacle in the rear changes its path to avoid colliding with the slow obstacle in the front. This is because the slow obstacle in the front cannot see the fast obstacle in the rear.

The specific rules used in the motion estimation for obstacles are as follows:

1) In case that a personal space of an obstacle is not violated, the motion of the obstacle is uniform linear motion.
2) In case that the personal space of obstacle is violated, the obstacle moves to the direction of the side of the obstacle to decrease the violation of its personal space until the personal space is not violated.

By imaginarily moving obstacles based on the above rules, a robot estimates future positions of obstacles.

IV. Conversion

In the second step, conversion, our method constructs a 3D time-space coordinate system from the estimated motions of the obstacles shown in section III.

At the beginning of this step, we define a *grid layer*, as shown in Fig. 3. In this layer, a 2D environment that is defined by the X and Y axis, is represented by grids. Each grid has a property, named *start grid* G_{start}, *goal grid* G_{goal}, *obstacle occupied grid* $G_{occupied}$ or *free grid* G_{free} that is not occupied. In addition, there are two types of G_{free}, that are *free-reachable grid* $G_{free-reach}$ that a robot can reach at the grid layer, and *free-unreachable grid* $G_{free-unreach}$ that the robot cannot reach.

Each grid layer is an expression of the 2D environment at certain times, because each grid layer has a time width T_{wid} (sec). These layers are stacked along the vertical axis based on the passage of time, as shown in Fig. 4, and called a *grid stack*. The bottom layer ($layer[0]$) represents the 2D environment from the present time (t=0 (sec)) to T_{wid} (sec) later of it. Likewise, $layer[n]$ represents the 2D environment from $t_{min}(n)$ to $t_{max}(n)$ where $t_{min}(n) = T_{wid} \times n$ and $t_{max}(n) = T_{wid} \times (n+1)$. The total stacking number of grid layers is represented as n_{layer}.

The occupied grids are defined by considering the personal spaces of all the obstacles in each layer, or each time period. Based on the above, the 3D time-space coordinate system is constructed by a voxel representation. Therefore, a grid G has three parameters, x_g, y_g and t_g. The x_g and y_g are parameters that specify the position of the grid in X-Y surface, and the t_g is a parameter to specifies the number of the grid layer that include itself. From now on, it should be kept in mind that a grid that has a property α is represented as $G_\alpha(x_g, y_g, t_g)$. In addition, a set of $G_\alpha(x_g, y_g, t_g)$ in the grid layer is represented as $Set_\alpha(t_g)$,

V. 3D path planning

In the third step, planning, a 3D path is planned in the 3D time-space coordinate system shown in section IV. Basically, to plan a path, we apply the distance transform algorithm proposed by Prof. Jarvis [4]. To apply it to our method, we extended the algorithm, as shown in the following.

A. Basic Distance Transform

First, we would like to explain a basic distance transform algorithm. A brief illustration of the idea is shown in Fig. 5. It is a grid based method where occupied grids and free grids are assumed to be predefined in advance.

In the first half of the algorithm, a distance field is generated by a distance function (Fig. 6). The value of the start grid is set to 0, and the values of the grids without obstacle occupied grids are calculated by a distance function, recursively, from the grids adjacent to the start grid. Therefore, after recursive calculations, each grid stores a grid-based distance from the start point to form the distance field. Although the distance function typically uses an approximate Euclidean distance, as shown in Fig. 6 (a), sometimes, for

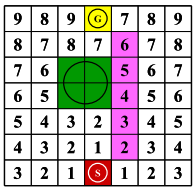
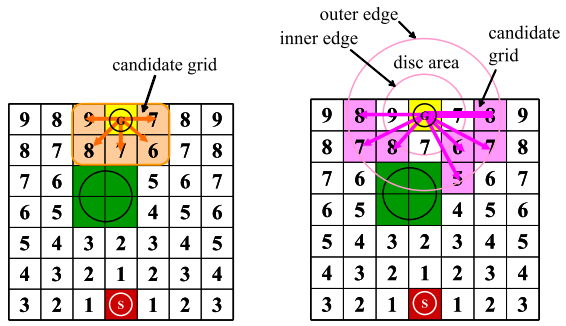


Fig. 5. Examples of 2D distance transform

(a) Approximate Euclidean dist. (b) City block dist.

Fig. 6. examples of distance functions



(a) Conventional method (b) Proposed method

Fig. 7. Difference between path search methods

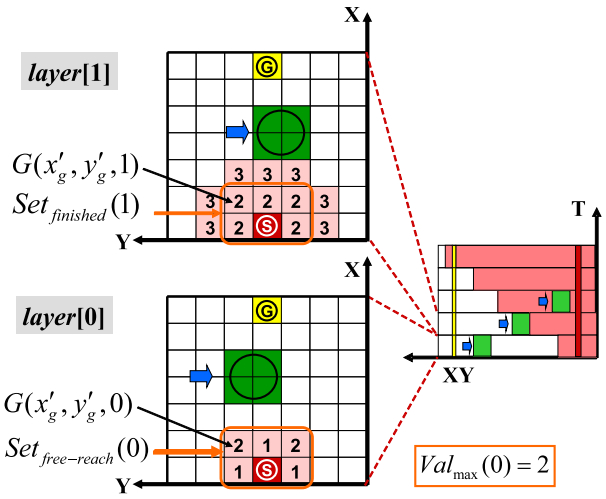


Fig. 8. The example of distance time transform

simplicity, an city block distance is used, as shown in Fig. 6 (b). Fig. 5,Fig. 7 and Fig. 8 show the latter example.

In the second half of the algorithm, a path is determined by descending the distance field from the goal grid. It is executed by successively moving to the grid that has the minimum distance value among the adjacent grids, from the goal grid to the start. In Fig. 5, , the pink grids represent the planned path from the start to the goal.

B. Extension of Path Search Algorithm

There is a problem with the original distance transform algorithm. In the second half of the algorithm, there are only 8 next grid candidates (Fig. 7-(a)). This means that the minimum angular resolution of the robot is 45 degrees. Furthermore the algorithm tends to choose diagonal grids from the goal grid, because its decreasing difference is larger than the horizontal and vertical differences. Therefore, it does not derive the geometrical shortest path.

To solve this problem, we improved the second half of the search algorithm to enhance the angle resolution of the path search. Fig. 7 introduces our basic idea. The idea is that the next grid candidates are not adjacent grids shown in Fig. 7-(a), but grids that centers of the grids are located on the search disc, shown in Fig. 7-(b).

The angle resolution of the path search is greatly affected by the internal radius R_{int} and external radius R_{ext} of the disc. A larger radius contributes a finer angle resolution to the path. However, it also increases the possibility of generating redundant path, because the length of each segment unit in the path becomes larger. In a case where the environment is not very complicated, we assign a value of 3.5 (grid) to R_{int} and a value of 4.5(grid) to R_{ext} in this research, based on our experience. We call this algorithm the *2D disc search algorithm*.

C. Distanc Time Transform and Extended 3D Path Search

1) Distance Time Transform: In the first half of the algorithm, in order to perform the distance transform algorithm in the 3D time-space coordinate system, we propose a distance time transform (DTT). The DTT is a procedure that discretely repeats distance transform from a present time to the future. In the 3D coordinate system, a grid layer represents the positions of obstacles and robot at the time period. Therefore, a set of free reachable grid is limited by the speed and current position of the robot.

The DTT repeats a subroutine that consists of three processes: “selecting,” “applying” and “copying.” Concretely, we execute as below.

At first, we focus on $layer[0]$ that represent the positions of obstacles and the robot from $T_{min}(0)$ to $T_{max}(0)$. Initially, the robot locates at G_{start}. The robot can reach to free grids that encircle G_{start}, that are $Set_{free-reach}(0)$. In a case where there are no obstacles in the grid layer, the shape of $Set_{free-reach}(0)$ is nearly a circle, and the size of the circle is determined by the speed of the robot.

We perform a distance transform in $Set_{free-reach}(0)$, and each grid of $Set_{free-reach}(0)$ stores a value of distance, called distance value. Then, a maximum distance value in $Set_{free-reach}(0)$, called $Val_{max}(0)$, is selected.

Next, a distance value of the grid $G(x'_g, y'_g, 0)$ in $Set_{free-reach}(0)$ is given to the grid $G(x'_g, y'_g, 1)$ if $G(x'_g, y'_g, 1)$ is not an obstacle occupied grid. After the all grids in $Set_{free-reach}(0)$ are copied by the above process, the distance values of the all grids in $layer[1]$ are changed to $Val_{max}(0)$

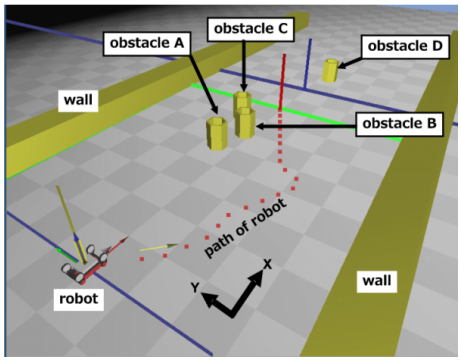


Fig. 9. Conditions for experiment in our simulator

for the initialization of performing a distance transform in $layer[1]$. A set of grids that store $Val_{max}(0)$ called *already distance transformed grids* $Set_{finished}(1)$.

Now, we focus on $layer[1]$, and select $Set_{free-reach}(1)$ that encircles $Set_{finished}(1)$. We also perform the distance transform, and copy the distance values in $layer[1]$ in the same way as in $layer[0]$. The DTT repeats the above subroutine until the number of focusing grid layer reaches the value of (n_{layer}-1).

2) Extended 3D Path Search: In the second half of the algorithm, to derive a 3D path in distance time transformed grid stack, we extend the *2D disc search algorithm* to the 3D space, named a *3D disc search algorithm*. First, we select $layer[m]$ that includes G_{goal} in $Set_{free-reach}(m)$. The path is determined by descending the discrete 3D distance field from G_{goal} in $layer[m]$. It is executed by moving to the grid successively that has the minimum distance value in the disc of the *2D disc search algorithm* in the grid layer $layer[k]$ (where $k = m-1$). In a case where there are no grids to be moved in $layer[k]$, it is executed in same grid layer $layer[m]$ in the same way as in *2D disc search algorithm*.

VI. Simulation Study

A. Mobile Robot Simulator

To confirm the applicability of the proposed method, we use the mobile robot simulator that we developed. A feature of this simulator is that a navigation program that works in the simulator is source-level compatible with our real robot, Kenaf. Therefore, we can confirm our navigation programs beforehand, but it does not guarantee the same motion in the real world because it does not consider the dynamics of mobile robots. Another feature of the simulator is that we can set not only static obstacles, but also moving obstacles. The obstacles' information can be obtained by the simulated 2D-LIDAR (Laser Imaging Detection and Ranging) system mounted on the simulated mobile robot. However, in this simulation, the motion vectors of the obstacles are obtained from the simulator directly. Fig. 9 shows an example image from the simulator.

B. Implementation Parameters

We implemented our collision avoidance method as a navigation program of the robot Kenaf. Each grid layer, as shown in section IV, is 10 (m) in length (8 (m) in the front and 2 (m) in the rear) and 8 (m) in width (4 (m) on the left and 4 (m) on the right). The area of each grid is 10 (cm) square. To form the 3D time-space coordinate system, the time interval between adjacent layers, $T_{interval}$, is set to 3 (sec), and the number of layers, n_{layer}, is set to 10. The robot's maximum velocity V_{max} is set to 400 (mm/sec). Therefore, the robot can reach 12000 (mm) ($= T_{interval} \times n_{layer} \times V_{max}$). The planning loop of the proposed method is conducted in every 1 (sec).

C. Condition of Moving Obstacles

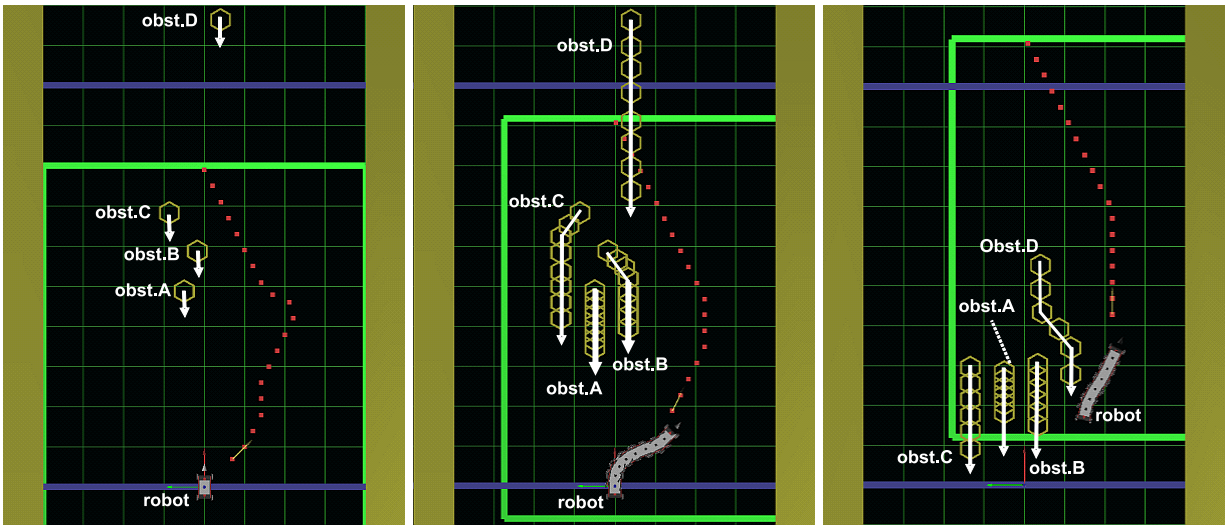
Fig. 9 shows a virtual walkway environment in the mobile robot simulator. The width of the walkway is 8 (m). We defined four moving obstacles in the simulator, represented by hexagonal cylinders, that move toward the robot along the walkway. The sizes of all the obstacles are the same, 30 (cm) in diameter. However, different values are used for the velocities, with obstacles A, B, C, and D moving at 20, 30, 40, and 50 (cm/sec), respectively. The motion rules for each obstacle are as follows: (1) it moves forward when there is no interference in front of it, and (2) it evades a certain distance when there are slower obstacles in front of it. The goal of the robot is located at the behind of the moving obstacles.

D. Simulation Result

We conducted several simulations using different initial positions for the obstacles. In some simulations, the robot avoided all the obstacles by estimating their movements. Figs.10 (a)-(c) show a good example where the proposed method worked well. In this figure, the red dots showed the planned path at the last time of the each figures with time width, the overlapping robots and obstacles show their respective trajectories that indicate their motions, the green boxes show the areas of each grid stack, and the white arrows on the obstacles' trajectories show their motions. The robot estimated the motion of obstacle D in advance; thus, it generated a detour path to avoid it.

To evaluate our method, we implemented a conventional collision avoidance method. Almost of all the path planning method is the same, but it does not estimate the future positions of moving obstacles while considering their personal spaces. Instead, it assumes that the obstacles move with uniform linear motion. In this method, obstacles may be imaginary overlapped (of course, this does not happen in the real world), but the robot does not consider it. Figs.10 (d)-(f) show the simulation results. The robot tried to move to the side of obstacle B, but later it collided with obstacle D while trying to evade obstacle B.

In this simulation, we did not consider occlusion of sensors. Furthermore, it is impossible to accurately estimate the obstacles' motions. Therefore, it is impossible to guarantee the effectiveness of the proposed method in the real world. However, the simulation results at least proved that the estimation of obstacle motions based on personal space worked reasonably well in comparison with the principle of uniform linear motion.



(a) t = 0 (b) t = 0 to t = 7 (c) t = 10 to t = 15

(d) t = 0 (e) t = 0 to t = 7 (f) t = 10 to t = 15

Fig. 10. Simulation results: ((a)-(c) results of our proposed method and (d)-(f) failure result of conventional method)

VII. CONCLUSION

We proposed a method for path planning without collision considering the motion of evacuees. The proposed method was implemented as the navigation program for a mobile robot, and some collision avoidance simulation experiments were conducted to verify its usefulness.

The future works are as below.

1) Separating the obstacles' motion estimation algorithm from the obstacles' motion decision algorithm
2) Consideration of the personal space of a robot
3) Consideration of complex motion rules of obstacles
4) Realization of real machine experiments (we have been developing a high speed 3D sensor that is able to detect 3D obstacles.)

REFERENCES

[1] T. Yoshida, K. Nagatani, E. Koyanagi, Y. Hada, K. Ohno, S. Maeyama, H. Akiyama, K. Yoshida, and S. Tadokoro, "Field experiment on multiple mobile robots conducted in an underground mall," *In Field and Service Robotics*, 2009.

[2] T. Tsubouchi, T. Naniwa, and S. Arimoto, "Planning and navigation by a mobile robot in the presence of multiple moving obstacles and their velocities," *Journal of Robotics and Mechatronics*, 1996.

[3] O. K. Gupta and R. A. Jarvis, "Optimal global path planning in time varying environments based on a cost evaluation function," in *Proc. of the 21st Australasian Joint Conf. on Artifi-cial Intelligence: Advances in Artificial Intelligence*, 2008.

[4] R. Jarvis and J. C. Byrne, "Robot navigation: Touching, seeing and knowing," in *Proc. of 1st Australian Conference on Artificial Intelligence*, 1986.

[5] R. Kirby, R. Simmons, and J. Forlizzi, "Companion: A constraint-optimizing method for person-acceptable navigation," in *The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 607 –612, 27 2009-oct. 2 2009.

[6] T. Amaoka, H. Laga, and M. Nakajima, "Modeling the personal space of virtual agents for behavior simulation," *International Conf. on Cyberworlds*, vol. 0, pp. 364–370, 2009.

[7] T. Osaragi, "Modeling of pedestrian behavior and its applications to spatial evaluation," in *AAMAS '04: Proc. of the Third International Joint Conf. on Autonomous Agents and Multiagent Systems*, 2004.