The 18th IEEE International Symposium on
Robot and Human Interactive Communication
Toyama, Japan, Sept. 27-Oct. 2, 2009

Real-Time Navigation for a Personal Mobility in an Environment with Pedestrians

Naotaka HATAO, Ryo HANAI, Kimitoshi YAMAZAKI and Masayuki INABA

***Abstract*— This paper describes a navigation system in a dynamic environment for a two-wheeled inverted pendulum mobile robot, PMR. Our system is organized by localization, detection and tracking of pedestrians, and trajectory planner. The localization is robust to effects of moving obstacles and pitching movements of the robot, and the trajectory planner creates a path with a certain smoothness considering movements of pedestrians. In addition, the planner introduces strategies to avoid pedestrians to be friendly to pedestrians around the robot. Besides, our system can run on two laptop PCs in real time. Finally, we show experimental results as well.**

I. INTRODUCTION

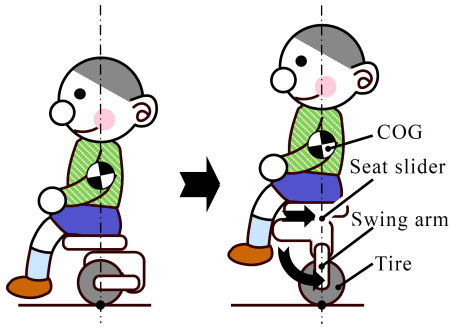
This work treats a pedestrian-friendly navigation system for a personal mobility. Personal mobilities mean single-seated mobile robots used for a short travel. To decrease the workload of the drivers, it is desirable for personal mobilities to be able to move automatically to their destinations. In that case, they must recognize pedestrians or other robots and avoid them to move safely.

Several robots which can move in the vicinity of people have been developed. For example, there are tour-guide robots typified by ”Minerva“[1], ”Blacky“[2], “TOURBOT“, and “WebFAIR“[3]. They can robustly estimate their coordinates in a crowded environment, and succeeded in autonomous navigation. Meanwhile, Hsu et al. developed a fast motion planning in a dynamic environment, and implemented it to an air-cushion robot[4]. This robot successfully reached the destination avoiding other robots which have a marker.

Our system can estimate positions and velocity vectors of pedestrians, and change the trajectory of the robot to avoid them. The trajectory planner consists of two different avoidance strategies. If there is enough space, the planner chooses a path to pass behind a pedestrian. If not, the planner chooses to wait until the pedestrian passes through. Besides, because our system can automatically distinguish moving obstacles from static obstacles with an LRF, it doesn't require any artificial landmarks or markers.

This paper is organized as follows: In section II, an overview of our approach and our robot “PMR” are described. The detail of the localization algorithm, detection and tracking of moving obstacles, and trajectory planning are shown in section III, IV, and V, respectively. The result of a navigation experiment in the real world is shown in section VI.

Fig. 1. Two-Wheeled Inverted Pendulum Mobile Robot “PMR”



(a) Gear down position (b) Gear up position

Fig. 2. The DoF of PMR

II. THE SYSTEM ARCHITECTURE FOR REAL TIME NAVIGATION

A. The Two-wheeled Inverted Pendulum Mobile Robot "PMR"

Fig.1 shows our mobile robot named “PMR”. It is a single-seated two-wheeled inverted pendulum mobile robot and originally was developed by Toyota Motor Corporation as “MOBIRO”.

Most of electric wheelchairs have four wheels, and the main defect of them is that they require large footprints. If one decreases the footprint of an electric wheelchair, the risk of falling down increased. In contrast, two-wheeled inverted pendulum mobile robots require in principle smaller footprints, because they can balance dynamically.

PMR has 5 DoF, that is, a seat slider, a pair of swing arms, and a pair of wheels. Before a driver rides on PMR, it becomes a gear-down position (See Fig.2 left). There are

978-1-4244-5081-7/09/$26.00 ©2009 IEEE

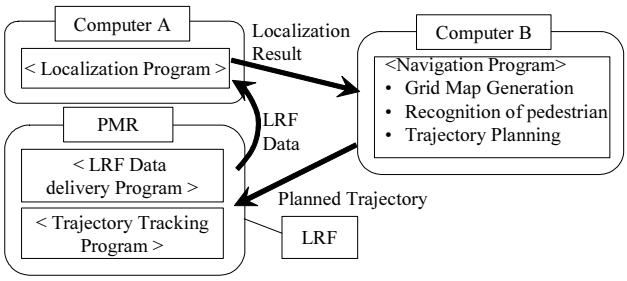


Fig. 3. The Navigation System Diagram

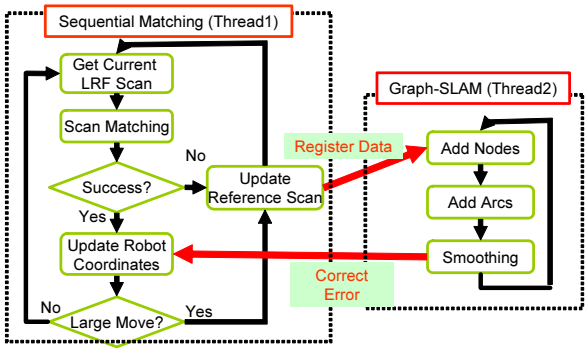


Fig. 4. The Architecture of Our SLAM Algorithm

two safety wheels (they are not shown in Fig.2) to prevent the robot from tipping over in the gear-down position. When moving, PMR becomes a gear-up position (See Fig.2 right). When PMR transits to gear-up position, the swing arms and the seat slider move to keep the position of COG. Besides, because the swing arms are independent from each other, even if one tire runs on a bump, the seat is kept horizontal.

PMR has several sensors such as LRFs and an omni-directional camera. In this research, we used one LRF (Hokuyo UTM-30LX) parallel to the ground in the gear-down position. The height of this LRF is around 1200[mm] in the gear-up state.

B. The Structure of the Navigation System

Fig.3 shows the diagram of the system architecture of the navigation software.

PMR has two laptop PCs in its body. Our system uses one of them for the localization, and the other for the trajectory planning. To avoid to lose the current coordinates of the robot, the localization software is always running. In contrast, the navigation software starts when required, and receives the results of the localization via TCP/IP socket.

III. LRF SLAM ROBUST TO MOVING OBSTACLES AND SWAYING MOTION OF THE ROBOT

A. The Overview of the SLAM Algorithm

In this section, our localization system using an LRF is described.

Moving people and shakiness of the vehicle body are the main two reasons which decrease the accuracy of SLAM for PMR. Many SLAM algorithms in a dynamic environment have been proposed[1][3][5][6]. Normally, these algorithms require to detect and track moving obstacles. In contrast, our algorithm does not explicitly detect moving obstacles, instead decreases the bad effects caused by moving obstacles and objects temporarily detected when the vehicle body leans.

In our algorithm, a sequential scan matching using ICP algorithm[7] is used as the first stage of SLAM. We improved the ICP algorithm to increase accuracy in dynamic environment. In addition, we adopted SLAM algorithm developed by Lu and Milios[8].

Lu and Milios's algorithm applies a graph structure to solve SLAM problem. The nodes of the graph are positions and rotations estimated by a sequential scan matching, and if the scan matching for two nodes succeed, an arc is made between them. Because the same scan matching algorithm as the sequential matching are used to make arcs of the graph, if the scan matching method is robust to moving obstacles and swaying motion of the robot, the whole SLAM algorithm also becomes robust.

Because Lu and Milios's algorithm is time-consuming, our implementation parallelizes the sequential scan matching and the execution of this algorithm(Fig.4). The main thread continuously executes the sequential scan matching. When the robot has moved large distance since the last node added, or the scan matching failed, a new node is added to the graph. In the latter case, the result of last succeeded scan matching is added. Once a new node is added to the graph, the second thread is activated, and executes the Lu and Milios's algorithm.

B. Estimation of Occluded Areas

Not only moving obstacles themselves but also occluded areas caused by them decrease the accuracy of ICP algorithm. It is easier to remove moving obstacles from LRF scan points than to remove occluded areas, because moving obstacles are normally detected as isolated clusters. In addition, even if the moving obstacle itself is small, the occluded areas become large when it passes near the LRF.

Our scan matching algorithm performs the estimation of occluded areas at every iteration of ICP algorithm, and uses the odometry of the robot and the sequential results of ICP algorithm to detect the occluded areas. These relative coordinates might be inaccurate initially, but become accurate gradually. Fig.5 is a conceptual diagram of detection of occluded areas. "M" means a moving obstacle, and "S" means a static obstacle. At first, the robot exists at R_1, then moves to R_2.

Occluded areas are detected as follows:

1) Find isolated clusters of LRF scan points. In Fig.5, scan points reflected by "M" and "S" objects are chosen.
2) Transform these scan points into R_2 coordinates. Our algorithm uses the odometry or the last result of ICP

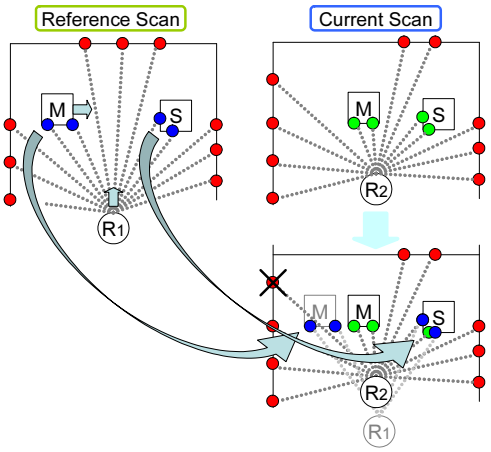


Fig. 5. The Detection of the Occluded Areas

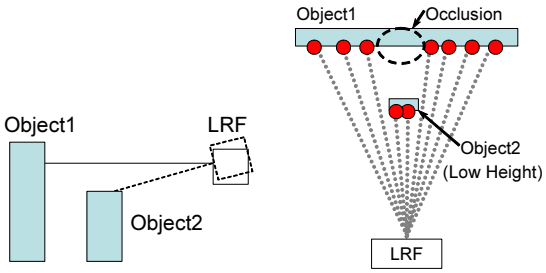


Fig. 6. Occlusion Area Caused by a Low-height Obstacle

matching as the relative coordinates between R_1 and R_2.

3) Place transformed points in scans obtained at R_2, and find the occluded area. In Fig.5, one scan point which might be occluded by "M" is detected. In the next iteration of ICP algorithm, this point is not used.

This detection method is effective to detect occluded areas caused by shakiness of the vehicle body. Fig.6 shows a typical situation occluded areas arise. In this case, however, our algorithm can detect the occluded area in the same way, because the scan points reflected by the Object 2 in Fig.6 are also isolated and bulged clusters.

Fig.7 shows the result of our SLAM algorithm. The length of the long side of the figure is $40.0[m]$. There were several pedestrians, but the SLAM succeeded. Fig.8 show the comparison between the result of our SLAM and the drawings of the floor, and the length of the long side of the figure is $53.0[m]$. The shapes of rooms and corridors were accurate.

IV. DETECTION AND TRACKING OF MULTIPLE PEDESTRIANS

Our system detects static and moving obstacles in a different way. At first, our system makes an occupancy grid map[9] which represents the positions of the static obstacles using a result of the localization mentioned above. Then, it

Fig. 7. The Result of SLAM in an Environment with Pedestrians

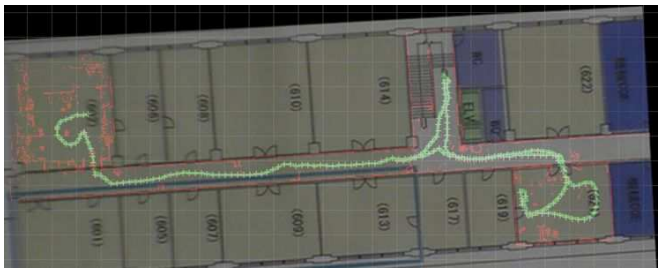


Fig. 8. The Comparison between the Result of SLAM and the Drawings of the Floor

estimates the positions and the velocity vectors of the moving obstacles (pedestrians). This section describes how to detect and track pedestrians.

A. The Estimation of Positions and Velocity Vectors of Pedestrians using Mixture Particle Filter

This section describes a method to detect and track pedestrians. Our algorithm approximates shapes of pedestrians by circular cylinders, and estimates their positions, velocity vectors, and radii using Mixture Particle Filter[10]. All LRF data and velocity vectors are transformed into the global coordinate with the result of the localization.

Normal particle filter is not suitable for tracking multiple targets, because particles are easily gathered around only one target. In contrast, Mixture Particle Filter can robustly track multiple targets, and can perform re-clustering of particles or merging and splitting of components without changing the distribution.

The parameters to estimate are the position (2 dimension), the velocity vector(2 dimension), and the radii of each pedestrian. So, the state of particles are 5 dimensions.

B. Discovering New Pedestrians and Abort of Tracking

Our algorithm uses a grid map mentioned before to detect pedestrians who started moving or appeared from blind spots of the LRF. If there is a cluster of LRF scan points which has appropriate size and does not overlap "occupied" grids, a new component added to the Mixture Particle Filter. "Occupied"

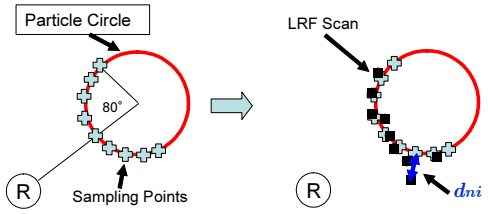


Fig. 9. The Calculation of likelihoods

in the grid map means that a static obstacle exists there, and moving obstacles should exist in "Free" grids.

In Addition, if a component enters a blind spot of the LRF or has low velocity for few second, it disappears. The latter means that a moving obstacle stopped and become a static obstacle.

C. Calculation of Likelihood

The height of the scan plane of the LRF equipped in PMR is $1200[mm]$, and it can scan the chests of pedestrians. Because, as mentioned above, the shapes of pedestrians are approximated by circular cylinders, the likelihood of each particle should be big if the sum of distances between the circle and LRF scan points is small.

The likelihoods are calculated as follows:

1) Obtain sampling points from each particle circle(See Fig.9). The center of the sampling point is placed on the intersection point of the circle and the line connecting the center of the circle with LRF, and the rest of them are placed at $10[deg]$ intervals on the circle.
2) The likelihood of ith particle is calculated as follows:

$$\sigma_i = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{\sum_{n=0}^{17} d_{ni}^2}{2\sigma^2}) \quad (1)$$

where d_{ni} is the minimum distance between nth sampling point of ith particle and LRF scan points, and σ is a coefficient.

The result of pedestrian detection and tracking is shown in Fig.10. The white rectangle is the center of the robot, colored clusters are the particles, white circles are estimated the position of the pedestrians, and the colored lines are the velocity vectors of them. Our algorithm could detect several pedestrians correctly.

V. TRAJECTORY PLANNING ALGORITHM USING CIRCLES AND TANGENT LINES FOR AN ENVIRONMENT WITH PEDESTRIANS

A. Trajectory Planning in a Dynamic Environment

The trajectory planning problem in the dynamic environment is an essential function for autonomous mobile robots, and have been under investigation for many years. Planning methods in a dynamic environment can be classified into three methods: deterministic, random, and reactive methods.

Most of the "Deterministic" approaches make a discretized configuration space, and solve it as the graph search problem

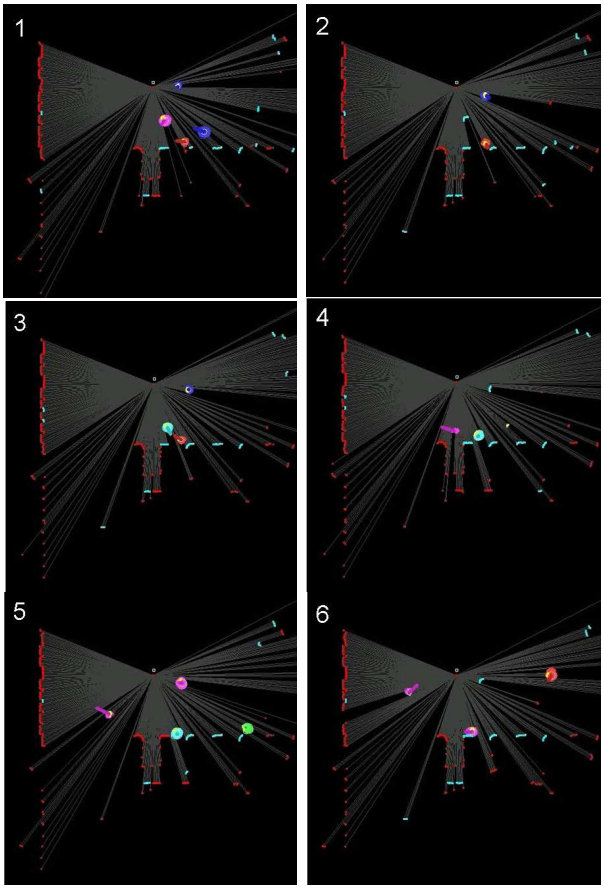


Fig. 10. The Result of Pedestrian Detection and Tracking

using Dijkstra, A*, D*[11], and so on. Although these methods assure that the shortest path is certainly found, the problem becomes very complex in dynamic environments, because the configuration space must include the time dimension. Fraichard proposed "state-time space" approach[12]. In this method, a path has been already given to a robot, and the robot avoids moving obstacles using only acceleration and deceleration. The method of [13] previously makes a smooth roadmap which a car-like robot can trace considering only static obstacles, then makes "state-time space" to avoid moving obstacles. Their methods realize short computational time, but are not able to make a path which avoids the moving obstacles.

Typical "Random" approaches are Probablistic Road Map (PRM) method[14] and Rapidly-exploring Random Tree (RRT) method[15]. Hsu el. al. modified PRM method to satisfy kinodynamic constraints, and succeeded in the trajectory planning in the real world using an air-cushion type robot[4]. The drawback of these methods is that the path becomes bendy and unnatural. Although several path smoothing methods were proposed[16], it is difficult to apply these methods to path planning in dynamic environments, because the estimated arrival time at each point of the path might change by these smoothing methods.

"Reactive" approaches determine the motion of robots

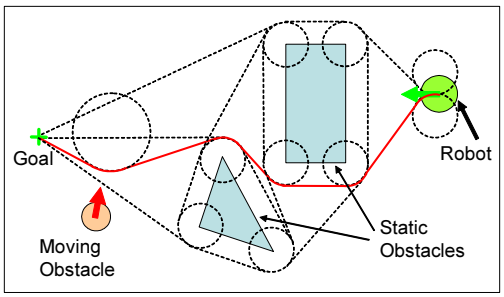


Fig. 11. The Conceptual Diagram of the Trajectory Planning Method

using current sensor input. Typical approaches are Dynamic Window Approach[17] and Potential Field[18]. These methods require very small computational time, but it is impossible to assure to reach the goal correctly. Several methods proposed which previously make a global path using known or static obstacles and deform the path to avoid unknown or dynamic obstacles[19][20].

Our approach is classified into deterministic methods. Random methods is not favorable for our robot, because, if the trajectory of the robot are bendy or includes many accelerations and decelerations, it will make the driver of the robot uncomfortable, and make pedestrians uneasy. Reactive methods is not favorable either, because the reaction velocity of PMR is relatively slow, and a safe path should be made well in advance.

The major drawback of deterministic methods is the long computational time caused by their vast search space. The path made by our approach consists of circular arcs and tangent lines, and the nodes which are used to solve the problem are the tangent points of them. In this scheme, the number of nodes become much smaller than in a uniform grid search space. Besides, since our approach doesn't need to smooth the path, it is possible to expect the arrival time at any point of the path. This means that our approach can expect the collision between the robot and the moving obstacles. To avoid moving obstacles, it can choose two options: one is to make a new trajectory which move behind the moving obstacle, the other is to wait until the moving obstacle passes through.

B. Trajectory Planning using Circles and Tangent Lines

Fig.11 is a conceptual diagram of our planning method. The candidates of a path are circles and tangent lines which connects these circles. The circles are placed at all vertexes of static obstacles and both sides of the center of the robot. The path are planned using A* algorithm. The nodes of a graph for A* are tangent points between a line and a circle, and the arcs are lines or circular arcs. The costs of arcs are the times that the robot takes to move from one side of the arcs to the other.

The robot traces the trajectory at a constant velocity if at all possible. The radius of circles must be larger than the radius of the robot. Besides, because our method uses a grid map to represent static obstacles, the circles which shares the center with occupied grids are placed. To decrease the number of circles, our method removes grids buried by other grids.

Because the costs are times instead of distances, our method can expect whether the robot and the moving obstacles collide with each other. If the robot collide with a moving obstacle in an arc, this arc are disabled. Alternatively, two avoidance strategies are employed: one is to make the new trajectory which move behind the moving obstacle, another is to wait until the moving obstacle path through. The former requires shorter time and larger free space to avoid the moving obstacle then the latter. Our methods does not make trajectories which requires acceleration to cut across in front of moving obstacles, because most moving obstacles are pedestrians, and they will be astonished if a large robot like PMR cut across in front of them at high speed.

In the real space, pedestrians often change the velocity or the direction, and it is impossible to expect the accurate future positions of them. However, it is not desirable the path changes frequently. So, our method execute re-planning only the current trajectory collides with static or moving obstacles, and the check of the collision is executed at short time intervals.

The overview of our methods are following.

1) Make concentric circles which shares the centers with the occupied grids. The robot can go around a circle clockwise and counterclockwise. For the sake of convenience, if two circle share the center and rotation directions are different from each other, they are treated as different two circles.
2) Make circles which the robot trace first using the current coordinate and velocity vector of the robot.
3) Make the first node using the current coordinates of robot, and add it to the Open Node List. The path-cost of node n ($h(n)$) is the estimated time of arrival to node n, and the heuristic cost of node n ($g(n)$) is the minimum time to the goal from node n. Therefore, distance-plus-cost of the node n ($f(n)$) is the sum of these two costs. In addition, our algorithm requires the coordinates(x, y, θ) and velocity of the robot at node n.
4) If there is no node in the Open Node List, the algorithm fails.
5) Choose and remove node n which has the minimum $f(n)$ cost from the Open Node List.
6) If node n is the goal node, the algorithm succeeds.
7) Make tangent lines between the node n with the static obstacle circles.
8) Choose lines which do not collide with other static and moving obstacles, and calculate the expected times of arrival to the another intersection of the line. Then, make nodes from these intersections and add them to the Open Node List.
9) Select lines which collide against dynamic obstacles, and calculate the expected times of arrival to the another intersection of the line. The expected times of

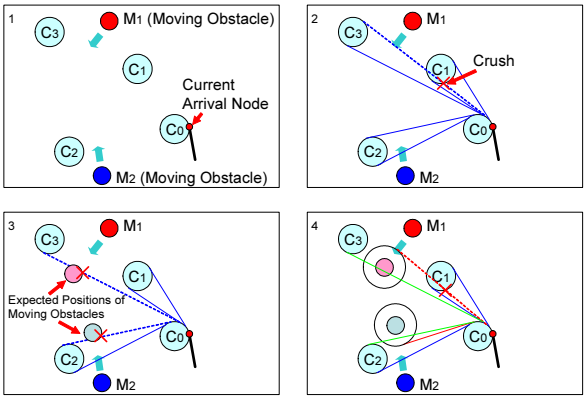


Fig. 12. The Conceptual Diagram of One Step of the Trajectory Planning Method

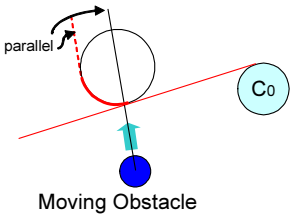


Fig. 13. The Conceptual Diagram of Avoidance of a Moving Obstacle

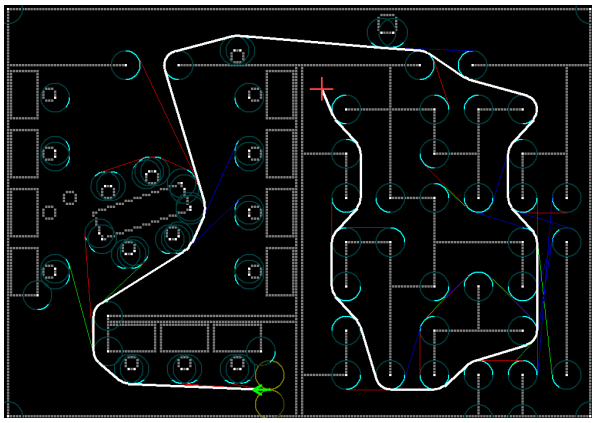


Fig. 14. The Result of the Trajectory Planning in a Static Environment

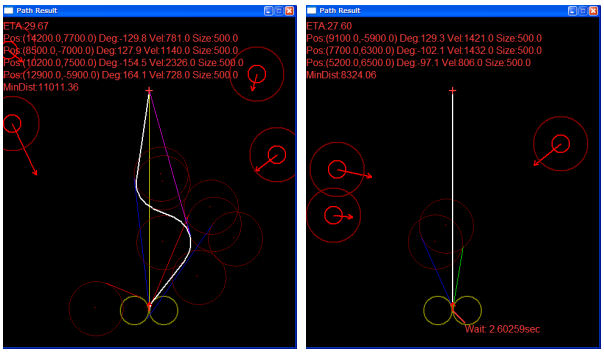


Fig. 15. The Avoidance of Several Moving Obstacles

arrival include the wait time of the moving obstacle. Then, make nodes from these intersections and add them to the Open Node List.

10) Make avoidance circles of moving obstacles, and make tangent lines between the current circle and the avoidance circles.
11) In the same way as 8), choose lines which do not collide with other obstacle, and make nodes from these intersections and add them to the Open Node List.
12) Return to 4).

Fig.12 is the conceptual diagram of one step of the algorithm. There are 4 static obstacles and 2 moving obstacles, and the current node n is on the circle C_0. Fig.12-2 and 3 correspond to 7) and 8). 6 lines which connects C_0 between other 3 static obstacles are made. But a line collides with a static obstacle, and two lines collide with moving obstacles. Fig.12-4 correspond to 9). The trajectory to avoid M_1 is invalid, because it collides with other static obstacles.

1) Avoidance of Moving Obstacles: Fig.13 is a conceptual diagram of how to make an avoidance circle of a moving obstacle. We define the sum of the radius of the robot, the radius of the moving obstacle, and a constant value for the safety as "Safety Distance". The distance of the robot between a moving obstacle must become longer than "Safety Distance".

First, make the line in which the minimum distance between the robot and the moving obstacle are the radius of the avoidance circle(the red line in Fig.13). Because it is difficult to determine this line analytically, our algorithm uses the bisection method. Then, place the circle to which the line is tangent, and whose center are on the extended line of the velocity vector of the moving obstacle. The radius of this circle is same as "Safety Distance". And, the red arc is the valid region of the circle.

This avoidance trajectory is not the fastest one, but because these are also represented by circles and lines, our algorithm treat them in the same way as the trajectories to avoid the static obstacles.

C. The result of the trajectory planning in the simulator

Fig.14-16 shows the results of the trajectory planning. Blue circles represent the static circles, the bold red circles represent the moving obstacle, the radius of the thin red circles which shares the center with bold ones means the "Safety Distance",the yellow circles represent the circle which the robot trace first, and the white lines are the planned trajectories.

Fig.14 is the result in a static environment. The shape of this environment is complicated, but the trajectory are correctly planned.

Fig.15 shows the avoidance of several moving obstacle. In the left case two avoidance circles of moving obstacles are made. In contrast, in the right case the planner chose to wait until the moving obstacles pass through. It is because

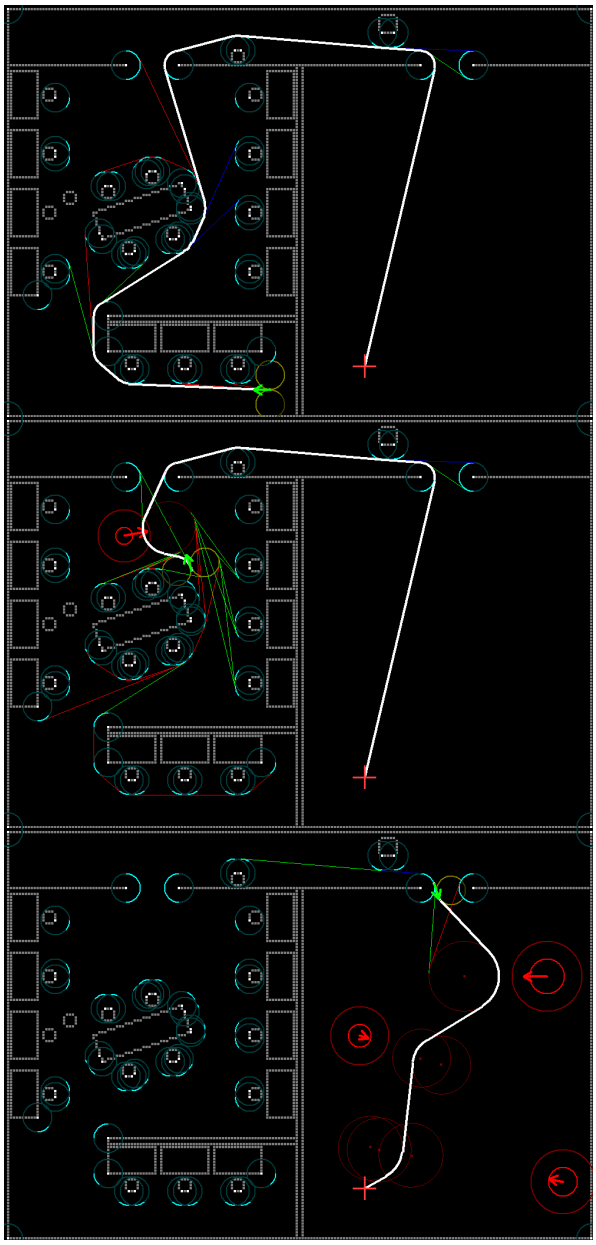


Fig. 16. The Result of the Trajectory Planning in a Dynamic Environment

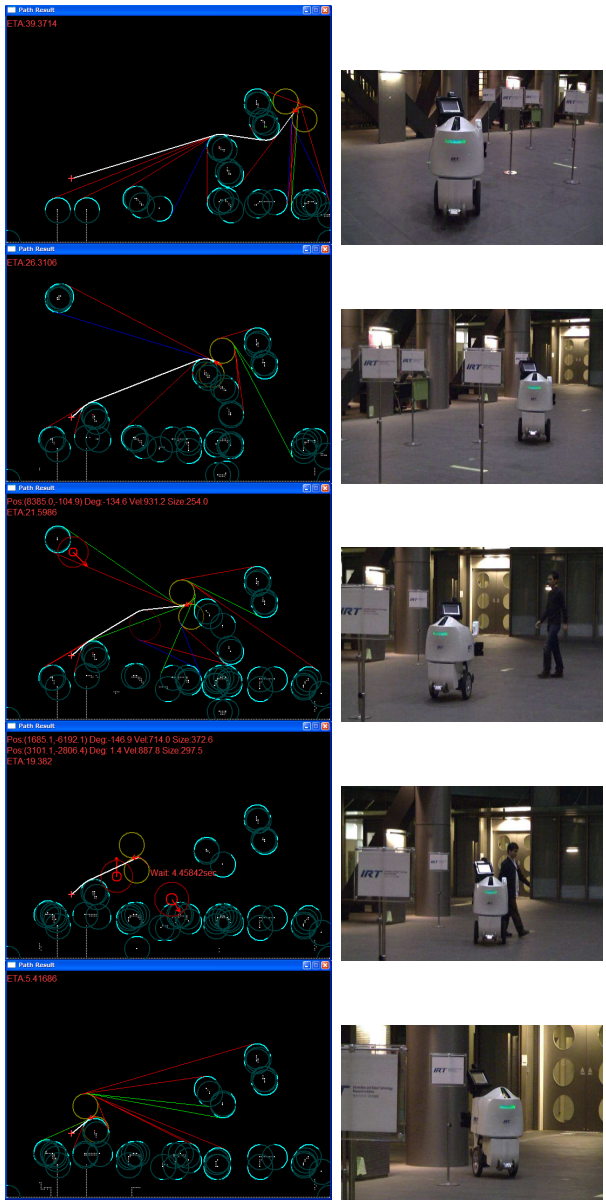


Fig. 17. The Result of Navigation Experiment

the trajectories to avoid moving obstacles collide with other obstacle.

Fig.14 is the result in a dynamic environment. Several moving obstacles were found while moving, but the replannings occurred correctly.

VI. NAVIGATION EXPERIMENT IN THE REAL WORLD

Fig.17 shows the result of the navigation experiment using PMR. The position of the goal was already obtained. The estimation of pedestrians and update of the grid map were executed in every $200[ms]$, and when a moving or static obstacle collides with the current trajectory, the re-planning is occurred. The velocity of PMR was $400[mm/s]$, the size of the grid map was $2000[mm] \times 1400[mm]$, the size of one grid was $100[mm]$, and the size of the static circles were $800[mm]$.

PMR avoided two pedestrians while moving. The planner made a path to avoid the first pedestrian(Fig.17-3), but it chose to wait until the second pedestrian pass through(Fig.17-4). It is because the second pedestrian appeared near static obstacles, and there was no enough space to make an avoidance circle.

VII. CONCLUSION

This paper described the navigation system in an environment with pedestrians. This algorithm was implemented to PMR, and it successfully and safely was able to reach the goal in an environment which pedestrians were walking around. Although PMR has only two laptop PCs, our system

could execute localization, estimation of positions of moving and static obstacles, and trajectory planning in real time.

Our current system uses only a horizontal LRF, and is not able to detect obstacles shorter than the height of it. In future, we are planning to adapt our system to 3D space.

ACKNOWLEDGE

This research is partly supported by Special Coordination Funds for Promoting Science and Technology, "IRT Foundation to Support Man and Aging Society".

REFERENCES

[1] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Minerva: A second-generation museum tour-guide robot. In *In Proc. of IEEE International Conference on Robotics and Automation (ICRA'99)*, 1999.

[2] D. Rodriguez-Losada, F. Matia, R. Galan, and A. Jimenez. Blacky, an interactive mobile robot at a trade fair. In *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA'02)*, 2002.

[3] P. Trahanias, W. Burgard, A. Argyros, D. Hahnel, H. Baltzakis, P. Pfaff, and C. Stachniss. TOURBOT and WebFAIR: Web-operated mobile robots for tele-presence in populated exhibitions. *IEEE Robotics and Automation Magazine*, 12(2):77–89, 2005.

[4] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233–255, 2002.

[5] D. Hahnel and D. Schulz and and W. Burgard. Mobile robot mapping in populated environments. *Journal of the Robotics Society of Japan*, 17(7):579–598, 2003.

[6] C. Wang and C. Thorpe and S. Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. *Proc. of Intenational Conference on Robotics and Automation(ICRA'03)*, 2003.

[7] P.J. Besl and N.D. McKay. A method for registration of 3D shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):239–256, 1992.

[8] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.

[9] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic robotics, 2001.

[10] J. Vermaak and A. Doucet. Maintaining multi-modality through mixture. *Proc. of 9th IEEE International Conference on Computer Vision(ICCV'03)*, pages 1110–1116, 2003.

[11] The focussed D* algorithm for real-time replanning. In *In Proc. of the International Joint Conference on Artificial Intelligence*, pages 1652–1659, 1995.

[12] T. Fraichard. Trajectory planning in a dynamic workspace: a 'statetime space' approach. *Advanced Robotics*, 13(1):75–94, 1999.

[13] J. van den Berg, M. Overmars. Kinodynamic motion planning on roadmaps in dynamic environments. pages 4253–4258, 2007.

[14] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[15] J. Kuffner and S. LaValle. RRT-connect : An efficient approach to single-query path planning. *Proc. of IEEE International Conference on Robotics and Automation (ICRA2000)*, pages 995–1001, 2000.

[16] J. C. Latombe. Robot motion planning. *Kluwer Academic Publishers*, 1991.

[17] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, 1997.

[18] O. Khatib. Real-time obstacle avoidance for robot manipulatorand mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[19] An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2002)*, pages 508–513, 2002.

[20] V. Delsart and T. Fraichard. Navigating dynamic environments using trajectory deformation. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS2008)*, pages 227–233, 2008.