2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

3D Time-space Path Planning Algorithm in Dynamic Environment Utilizing Arrival Time Field and Heuristically Randomized Tree

Igi Ardiyanto and Jun Miura

Abstract— **This paper deals with a path planning problem in the dynamic and cluttered environments. The presence of moving obstacles and kinodynamic constraints of the robot increases the complexity of path planning problem. We model the environment and motion of dynamic obstacles in *3D time-space*. We propose the utilization of *the arrival time field* for examining the most promising area in those *obstacles-occupied* 3D time-space for approaching the goal. The arrival time field is used for guiding the expansion of a randomized tree search in a favorable way, considering kinodynamic constraints of the robot. The quality and the optimality of the path are taken into account by performing heuristic methods on the randomized tree. Simulation results are also provided to prove the feasibility, possibility, and effectiveness of our algorithm.**

I. Introduction

Producing an efficient trajectory for the robot in the cluttered and unknown environment is one of complicated path planning problems. The complexity will increase along with the presence of dynamic obstacles. For example, when a dynamic obstacle passes in front of the robot which is moving toward a goal, there are several options for the robot to act. The robot can take the route which is predicted to be left by the dynamic obstacle, or the robot takes a way which is predicted not to be reached by the obstacle in the future, or the robot can take the straight path by a stop-and-go movement. In this problem, determining the most optimal route for the robot is a difficult task.

The complexity of the problem will also increase by adding the number of dynamic obstacles. In the classic path planner, this problem is solved by doing a collision check algorithm, but such a method does not take account the quality and the optimality of the path. In several recent approaches, handling the moving obstacles is done by using the time-space approach. The use of time-space representation has several advantages; one of them is that we have a wider view for seeing the moving obstacles than using only spatial approach, so that we can do more than just for avoiding them.

Another important thing when we develop a path planning algorithm is how the path planner can be applied to real robots and environments. We have to consider kinodynamic constraints of the robot. Besides that, the execution time of the algorithm becomes a critical matter.

In this paper, we propose a novel approach dealing with these problems by introducing the *arrival time field* to guide a randomized path search in a 3D time-space. By looking at the problem using the time-space approach, we can use the information of moving obstacles, not only for avoiding them but also to examine regions between obstacles and to get the most efficient region for the robot to move toward the goal. In our approach, basically, the introduced arrival time field is used to provide information about those efficient regions which can ensure the convergence and the optimality of the path, and then combined with a randomized search method which can widely explore the area. The utilization of randomized search is also useful for considering kinodynamic constraints and the path quality of the robot, so that the produced algorithm is applicable to the real robot.

The rest of this paper is organized as follows. We present related works in section II, followed by the basic idea of the arrival time field and a brief explanation of the arrival time field in 3D time-space and its properties in section III. Section IV describes the randomized tree and heuristic methods to optimize generated path. We then show the simulation results in section V. Conclusion of our work and possible future work are described in section VI.

II. Related Works

There are a lot of works which have been presented and discussed to address the problem of path planning. Randomized technique is one among many approaches which is used by many researchers ([6], [4]). Randomized path planner such as RRT (Rapidly-exploring Random Tree) [2] is widely accepted because of its ability to explore the tree in the vast area. Several researchers have been worked on this sampling-based method and its variant, like Jaillet *et al*. [6], Zucker *et al*. [7], Karaman and Frazzoli [5],Urmson and Simmons [3], and Vonasek *et al*. [15]. There are also researches on RRT using kinodynamic constraints by La Valle, *et al*. [4] and Plaku, *et al*. [8]. Another work proposed by Hassouna *et al*. [1] does not use randomized technique, but instead uses a potential function generated by the Level Set Method over the free space.

Other researchers try to solve the problem of moving obstacles by using time-space approach due to its advantages as we have mentioned in the previous section. Fujimura [17] has employed the time-space environment to generate an optimal motion, with a restriction that the obstacle motion is completely known. Zucker, *et al* [16] develop a variant of RRT algorithm for system with time-space constraints. The algorithm supports planning in unknown environments, but it does not consider kinodynamic constraints of the robot.

I. Ardiyanto and J. Miura are with Department of Computer Science and Engineering, Toyohashi University of Technology, 1-1 Hibarigaoka, Tenpaku-cho, Toyohashi, Aichi, 441-8580, Japan {iardiyanto, jun}@aisl.cs.tut.ac.jp

III. Arrival Time Field in 3D Time-space

Before we talk further about the arrival time field in 3D time-space, we first describe the basic idea of the arrival time field itself in the lower dimension. Let us consider the environment $\mathbb{C}$ as a two dimensional time-space that holds information as follows:

- Non-passable area, holds information about walls and static obstacles, denoted by $\boldsymbol{O} \subseteq \mathbb{C}$;
- Free space area, defines visible areas where robot will not collide walls as well as static obstacles, denoted by $\boldsymbol{F} \subseteq \mathbb{C}$;

A. Definition

We define the *arrival time field* as a space containing the information about the time needed by each point in the space for approaching a determined point. Let us first consider basic kinematic equation in one dimensional case which correlates speed V, position x, and time T

$$\Delta x = V \Delta T \Longrightarrow \frac{\Delta T}{\Delta x} = \frac{1}{V} \tag{1}$$

In higher dimension, eq. (1) can be expressed as

$$|\nabla T| = \frac{1}{V} \tag{2}$$

Let a monotonic wave front originated from a determined source point move across a space. Then the arrival time of wave front in every point can be calculated using (2). The arrival time of a point depends on the distance from the source point and the speed used for travelling the wave front toward that point. This problem is known as *Eikonal equation* problem which can be solved by Godunov appoximation [11], which is for example in a 2D space

$$\sqrt{\max\left(D_{i,j}^{-x}T, -D_{i,j}^{+x}T, 0\right)^2 + \max\left(D_{i,j}^{-y}T, -D_{i,j}^{+y}T, 0\right)^2} = \frac{1}{V_{i,j}}; (i,j) \in \boldsymbol{F} \tag{3}$$

where

$$\begin{aligned} D_{i,j}^{+x} &= T_{i+1,j} - T_{i,j}, D_{i,j}^{-x} = T_{i,j} - T_{i-1,j}, \\ D_{i,j}^{+y} &= T_{i,j+1} - T_{i,j}, D_{i,j}^{-y} = T_{i,j} - T_{i,j-1}. \end{aligned} \tag{4}$$

$T_{i,j}$ is the arrival time value of cell (*i,j*), and $V_{i,j}$ denotes speed function of cell (*i,j*). Solution of (3) can be retrieved using the solver such as Fast Marching Method [11], Fast Sweeping Method [12], or Fast Iterative Method [14].

The stressing point of the arrival time field is that it provides the minimum predicted arrival time for each point rather than the shortest distance to the goal. This predicted arrival time depends on its speed function. We can exploit the speed function by inserting information about several possibilities that the robot may face in the environment, such as rough terrain, variational distance of obstacles, etc.

For example on a plain environment with obstacle, we can determine that the robot is better to move slower at the narrow space or corridor or area next to the obstacle. We then set a smaller speed on the area near the obstacle (for example, using monotonic velocity function which gives more value of speed at an area far from the obstacle), then a shorter travel time will be through an area far from the obstacle (see Fig. 1). It means the safety factor is also taken into account in the arrival time field calculation. For that purpose, we implement a monotonic function denoted by

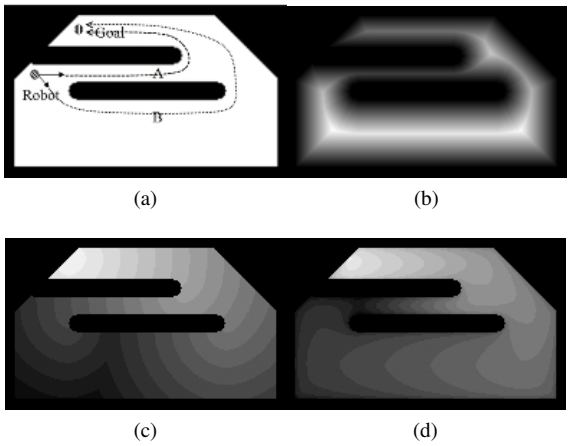


Fig. 1: (a) An environment with two possible paths towards the goal. (b) Monotonical velocity field of (a). (c) Arrival time field with a uniform velocity function. (d) Arrival time field with a monotonic velocity function. Darker color means longer arrival time. By using the monotonic velocity function, taking path B is better according to (d)

$$V_{i_1,j_1} = \begin{cases} n^{\left\| x_{i_1,j_1} - x_{i_2,j_2} \right\|} & \text{for } x_{i_1,j_1} \in \boldsymbol{F}, x_{i_2,j_2} \in \boldsymbol{O} \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

where V_{i_1,j_1} is the speed on the point x_{i_1,j_1}, x_{i_2,j_2} is the nearest point of obstacle to x_{i_1,j_1}, and n is a constant for adjusting the monotonic function's value, to give more differences on each cell.

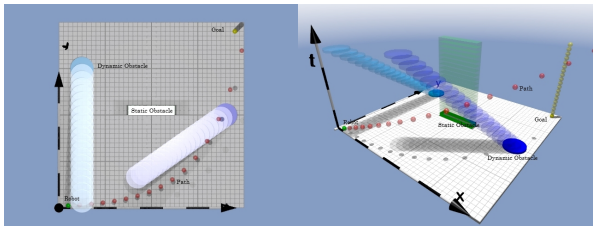
B. Modelling The Environment

We have described the arrival time field and have given the example in 2D environment. In our proposed algorithm, we use a 3D time-space representation. Basically, the modeled environments are consist of collection of predicted 2D environments in several time slices. In each time slice, we estimate the position of every object such as static obstacles, walls, and moving obstacles (see Fig. 2). For dynamic obstacles, we make a short-time dynamic obstacle motion model using constant speed for motion prediction of dynamic objects. Let $Z_x^{'}(t)$ and $Z_y^{'}(t)$ be the predicted position of an obstacle at time slice t in x and y coordinate. We predict the position of each dynamic obstacle by

$$Z_x^{'}(t) = Z_x(0) + v_{Z_x} t, \qquad Z_y^{'}(t) = Z_y(0) + v_{Z_y} t, \tag{6}$$

where $Z_x(0)$ and $Z_y(0)$ are the current position of each obstacle, and v_{Z_x} and v_{Z_y} are the speed of the obstacle on the respective x and y coordinate.

We assume that motion prediction of moving obstacle is only effective for a short range of time, due to its uncertain behavior. We currently use fixed 10 time slices with cycle time of 1000 milliseconds for modelling the dynamic



(a) (b)

Fig. 2: Modelling the environment, (a) 2D model, (b) 3D time-space model

obstacles, started from the time of the current state of the robot.

C. Algorithm of 3D Arrival Time Field

The basic purpose of the arrival time in the 3D time-space is to predict the arrival time of each point in every time slice, so that we will be able to determine which region has the most minimum arrival time. The result of arrival time calculation is suitable for the point robot, i.e. robot can freely move in all direction. Kinodynamic constraints of the real robot tends to make the travel time increase and make it difficult to predict the exact arrival time. For that reason, we use the arrival time field as a bias for guiding the randomized tree considering kinodynamic constraints [13] rather than use it directly as the point robot movement.

We need to make several scenarios of arrival time describing the travel time from start to goal, beginning from the best case to worse cases. The example of best case is when the robot can directly take a straight path to the goal, while for the worst case, we limit it to 10 time slices longer than the best case. For each scenario, we propagate the wave front using our proposed algorithm, then we merge all of scenarios using the best value one.

Calculating the evolution of wave front using *partial differential equation (PDE)* solver such as Fast Marching Method (FMM) and Fast Iterative Method (FIM) is very costly, even for low dimensions. For solving PDE in the 3D time-space environment, it is impossible to use any existing method if we aim for an online path planning algorithm. We here propose a new method for solving those problems in the 3D time-space, by using *time boundaries variant* of Eikonal solver, which means the farthest area reaching by the wave front is increased along with the time slice (see Fig. 3). We extend the FIM (for basic explanation about FIM, please refer to [14]) to the 3D time-space environment. We maintain an *active list* (see Fig. 4) for each scenario, then we run those *active list* simultaneously (see algorithm 1, 2, and 3). We take advantage of the behavior of FIM which will iteratively update each point until reaching convergence.

We predict the best scenario for the wave front to reach the robot from goal position as the minimum time slice, and the worst scenario given by

$$T_{best} = \frac{dist_{min}}{v_{max}} \quad and \quad T_{worst} = T_{best} + 10 \qquad (7)$$

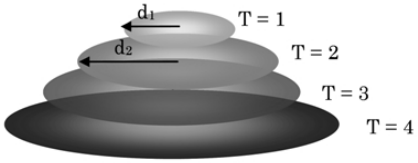


Fig. 3: Time boundaries for evolution of wave front, $T = 1$ means wave front can only grows as far as d_1 along first time slice and so on.

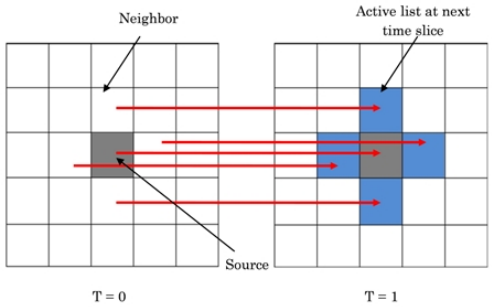


Fig. 4: One example for creating an active list to be used on the next time slice (see Algorithm 3).

Algorithm 1: Extended Fast Iterative Method (1)

X = set of points x in the map for all time slice t
L = active list

Function : Initialization
for all $x \in X$ **do**
  $x = \infty$
  for $t = T_{best}$ to $t = T_{worst}$ **do**
    if $x = goal$ **then**
      $x = 0$
      add neighbor of x to L
    end if
  end for
end for

Algorithm 2: Extended Fast Iterative Method (2)

th = very small value to determine the convergence of x

Function : Fast Iterative Method
while L is not empty **do**
  for all $x \in L$ **do**
    $\Delta x \leftarrow solve\_eikonal(x) - x$
    if $\Delta x \leq th$ **then**
      Update List (x)
      remove x from L
    end if
  end for
end while

Algorithm 3: Extended Fast Iterative Method (3)

Function : Update List (x)
if x in time slice T **then**
  add 4-neighbor of x in T+1 to L
end if

where T_{best} is the arrival time at the best situation, T_{worst} is that at the worst situation, $dist_{min}$ is the Euclidean distance from the current position to the goal, and v_{max} is the maximum robot speed.

We modify the Eikonal equation solver in (4) to fit the 3D time-space environment. Eq. (4) becomes

$$\begin{aligned} D^{+x}_{i,j} &= T^{next\_time\_slice}_{i+1,j} - T^{current\_time\_slice}_{i,j}, \\ D^{-x}_{i,j} &= T^{current\_time\_slice}_{i,j} - T^{next\_time\_slice}_{i-1,j}, \\ D^{+y}_{i,j} &= T^{next\_time\_slice}_{i,j+1} - T^{current\_time\_slice}_{i,j}, and \\ D^{-y}_{i,j} &= T^{current\_time\_slice}_{i,j} - T^{next\_time\_slice}_{i,j-1}. \end{aligned} \tag{8}$$

IV. HEURISTICALLY RANDOMIZED TREE

A. Notation

Our randomized tree is constructed by collections of reachable states $\mathbb{S}$ called node. Every node is defined by the tuple $S = \{x, y, \theta, v, w, t\} \in \mathbb{S}$, representing the robot position in xy-axis and its heading θ, the current translational velocity (v) and the rotational velocity (w) of the robot in that node, and time t for reaching that node from the current state.

We give a predefined set of possible motions to the path planner. Each motion in the set consists of a translational and a rotational velocity as robot control notated by $u_i = \{v_i, w_i\}, (i = 1, 2, \ldots, K)$, where K is the total number of the motions, which satisfies kinematic constraints of the robot.

Let S_t be the current state and S_{t+1} be the next state reached from S_t using a chosen motion $u = \{v, w\}$. We define this action of extending state S_t as a function

$$S_{t+1} \leftarrow g(S_t, u). \tag{9}$$

B. Arrival Time-Biased Random Tree

We expand the tree from the current position of the robot, using a similar approach to HeAT Random Tree algorithm in [13]. We select a random point using the bias from the *arrival time field* so that the tree grows in a favorable direction towards the goal. We iteratively choose a random point S_{target} which has higher value of arrival time field than a threshold value. This threshold value is started from the value of the arrival time field of the initial state, and grows when a new created node has a better value than this threshold. We then choose the nearest node S_{near} to the S_{target} among all of nodes in the tree $\mathbb{S}$.

Every time S_{near} is chosen and eligible to be expanded, we will calculate a new state S_{new} of that node (S_{near}) by evaluating all of possible motion controls

$$S_{u_i} \leftarrow g(S_{near}, u_i), \text{for } i \in \{1, 2, 3, \ldots, K\} \tag{10}$$

where S_{u_i} is the extention of S_{near} using motion control u_i, and K is the total number of the motion set, which satisfies kinematic constraints of the robot. We do not need to perform collision checking further because the information about both static and dynamic obstacles have been embedded in the 3D time-space field itself.

We then pick the best motion control

$$u_{best} = \arg\min_i cost(S_{u_i}), \tag{11}$$

which gives the best cost function, to get the new node

$$S_{new} = S_{u_{best}} \leftarrow g(S_{near}, u_{best}), \tag{12}$$

$cost(S)$ is a cost function for evaluating a motion, defined by

$$\alpha M_1 + \beta M_2 + \delta M_3, \tag{13}$$

$$M_1 = bias(S), \tag{14}$$

$$M_2 = dist(S_{target} - S), \tag{15}$$

$$M_3 = |\theta_S - \theta_{S_{near}}, | \tag{16}$$

where $bias(S)$ is the arrival time value at predicted point(x_S, y_S), $dist((x_{S_{target}}, y_{S_{target}}) - (x_S, y_S))$ is the distance between destination point $(x_{S_{target}}, y_{S_{target}})$ and the predicted point (x_S, y_S). $|\theta_S - \theta_{S_{near}}|$ is the heading difference of the robot between the current state and the predicted state, and α, β, and δ are constants.

We need to determine a proper criterion for extending every node chosen by the randomized planner to reduce inefficient and disperse motions. In this case, we want to reduce unstable movements that are often found in the path created by randomized planner. We use the previous heading criterion as defined in (16) to ensure that we will not choose a very large difference of heading on each pair node causing unstable movements.

C. Fast Replanning Algorithm

We use a pretty fast time cycle (currently, 1000 milliseconds per cycle) for calculating the entire algorithm i.e., updating map information, static and dynamic obstacles, and performing calculation of 3D Arrival Time Field and Heuristically Randomized Tree. We assume that the environment is not so much changed during that cycle. The path generated by the previous calculation is still expected to be feasible for the current cycle. The previous path is examined from the root to the longest collision-free state of the path and used it as initial tree for the current calculation.

V. RESULT

A. Offline Simulation

In the offline simulation, we set the environment combining static and dynamic obstacles so that there are several routes for the robot to reach the goal. Three dynamic obstacles then are put in the fastest route (i.e. the middle route, see Fig. 5) and are run in several scenarios; completely block the route and move toward the robot, block the route but move in the opposite direction of first scenario, and give a possible way by moving one obstacle in the opposite direction. The environment consists of a 200x200 cells of map with the free space and static obstacles, where blue area denotes free space, green line is wall, the orange circle denotes robot position, the red circle is the goal, the triangles are moving obstacles, and the black area is extending space for obstacles.

TABLE I: Calculation cost of 2D and 3D Algorithm

Type of Calculation	2D Planner	3D Planner (ordinary FMM)	3D Planner (Ext. FIM)
Arrival Time Field calculation	86 ms	$\geq$100000 ms	400 ms
Random Tree calculation	350 ms	250 ms	250 ms
Total Time	500 ms	$\geq$100000 ms	1000 ms

TABLE II: Comparison of 2D and 3D Planning Algorithm

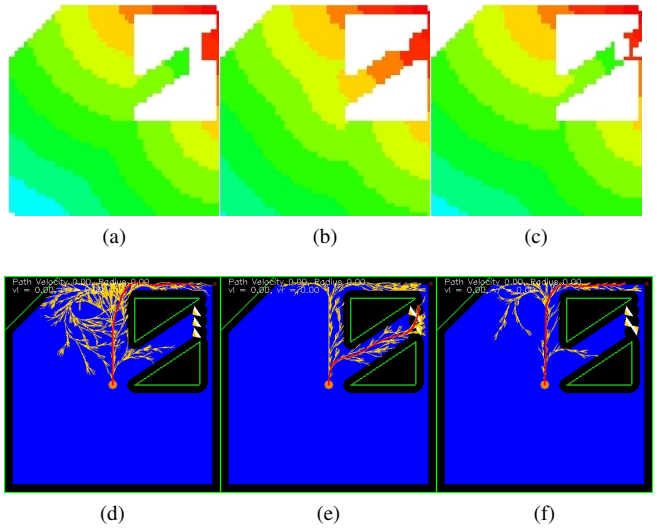
Statistic		Blocking Scen.	Crossing Scen.
Number of simulation		10 times	10 times
Averaged path cost (time-to-goal)	2D	35 sec	15 sec
	3D	26 sec	11 sec
Successful runs	2D	30%	70%
	3D	100%	100%
Averaged number of nodes	2D	500 nodes	1500 nodes
	3D	4000 nodes	3000 nodes

Another scenario is that there are several moving obstacles cross in front of the robot and block robot's way (see Fig. 6).

To check the performance of our 3D time-space path planner, we compare it with *collision check-based 2D path planner* [13]. This 2D path planner also uses the arrival time field as a guidance for growing the randomized tree. The main difference with the 3D time-space arrival time field used in our 3D path planner is that the 2D algorithm only considers static obstacles when it makes the arrival time field, and performs collision checks for each time we grow the tree to avoid the moving obstacles.

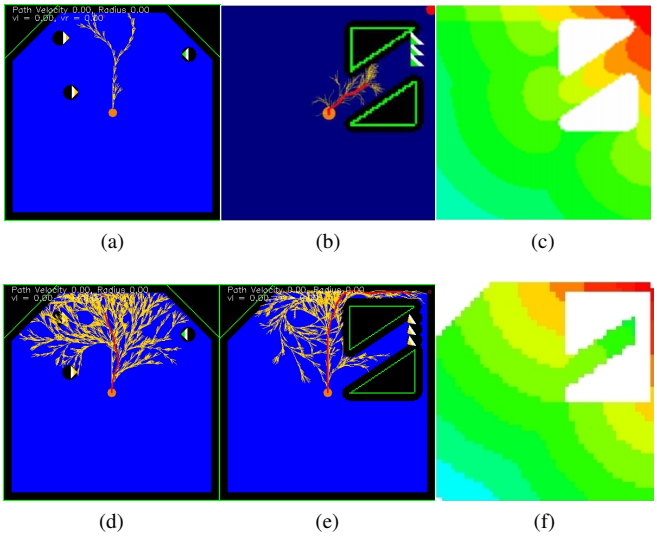
We compare the 3D time-space algorithm and collision check-based algorithm in 2D environment by applying the crossing obstacles scenario (Fig. 6a and 6d) and the blocking scenario (Fig. 6b and 6e) to distinguish and prove the effectivity of searching path in the 3D time-space environment. The 2D path planner algorithm will face difficulties if several moving obstacles are blocking the way, like Fig. 6b.

Table I shows us comparison of the calculation cost of the 2D and 3D path planner. It makes sense that the 3D algorithm takes more time than the 2D one. The most interesting part is that by using our extended FIM algorithm, we can dramatically reduce the calculation time of the wave front propagation in the 3D time-space, compared with ordinary sequential FMM algorithm. Even for the randomized tree calculation, the 2D planner takes more time than the 3D one, due to collision check used by the 2D planner, while the 3D planner does not need this action. So that, with a reasonable difference on calculation time wtih the 2D path planner, we still can manage to build an online 3D time-space path planner. Those differences on calculation time can be covered by the performance of 3D time-space path planner. Based on quantitative data (see table II), it is clear that 3D time-space algorithm outperforms the collision check-based algorithm in 2D environment.



(a) (b) (c) (d) (e) (f)

Fig. 5: Comparison of different scenarios on offline simulation (bottom row) and its 3D time-space arrival time field on time slice T = 5 (top row), a) and d) Moving obstacles are completely blocked the way and move toward the robot. b) and e) Two of obstacles are away and give enough space to the robot for moving. c) and f) Only one obstalce is away and the robot will choose safer way. The goal is at top-right corner.



(a) (b) (c) (d) (e) (f)

Fig. 6: Comparison of 2D approach (top row) and 3D time-space approach (bottom row) on: a) and d) crossing obstacle scenario, b) and e) blocking scenario, c) and f) arrival time field of blocking scenario.

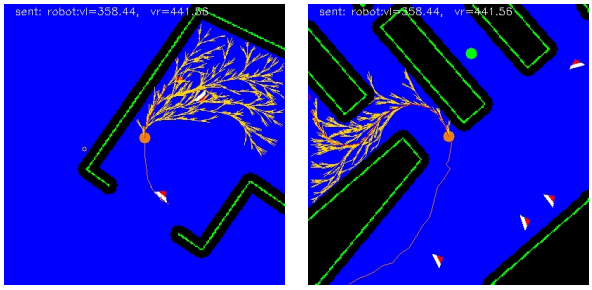
B. Online Simulation

We test 3D Time-space Random Tree path planner in a simulation representing the real robot and environment. All of implementations were done using a laptop PC (Core2Duo, 2.1 GHz, 2GB memory, Windows XP). We implement our path planner algorithm as an RT-component which is software module running on RT-middleware[1] environment [10] for reusability.

[1]RT-middleware is a specification on a component model and infrastructure services applicable to the domain of robotics software development, authorized by OMG (Object Management Group).

TABLE III: Statistic of Online Simulation Result

Statistic	Value
Arrival Time Field calculation	400 ms
Random Tree calculation	250 ms
Number of nodes	4000 nodes
Maximum speed	400 mm/second
Number of dynamic obstacles	5-12 objects
Total simulation time	600-900 seconds
Number of simulation	10 times
Successful runs	100%



(a) (b)

Fig. 7: Screenshot of simulation using Environment and People Movement Simulator

We use a simulator [9] to perform simulation of our path planner. The simulator generates a 200x200 cells of map consist of free space and static obstacles mimicking the canteen of our university. This simulator also provides dynamic objects information to the path planner, which represents the movement of people. We apply the 3D Time-space Random Tree path planner to people tracking problems (see Fig. 7).

The robot has to follow one of dynamic obstacles considered as a person while avoiding static obstacles, walls, and other people (around 5-12 persons) moving inside the simulator. Simulation's flow is as follows: people enter the canteen, queue for the ticket at ticket machine, take the meals, go to a free seat, stay on the seat for eating, bring the tray to the washing place, and go to the exit. The robot is considered having succeeded in solving the people tracking problem when the robot can follow the tracked person from the entrance to the exit.

Table III shows the robustness of our algorithm. All of 10 times simulations have been done successfully. Overall calculations need less than 1000 milliseconds, it means our 3D time-space planning algorithm can be run online.

VI. CONCLUSIONS

We have presented a novel real-time path planning algorithm, using *the arrival time field* in 3D time-space as bias for a randomized tree search. Heuristic approaches of our algorithm are proved to generate a smooth and safe path for the robot. Our proposed method for solving wave front propagation in the 3D time-space makes our path planner able to run online. Simulation results show that our algorithm is applicable to the real robot and effectively handle a dynamic environment and kinematic constraints of the robot.

Calculating the arrival time field on the 3D time-space takes the most time in our proposed algorithm. Several improvements are possible to be applied. One of them is to optimize the algorithm of 3D time-space arrival time field calculation, for example by parallelize it. Recent technologies on parallel computing give us a high possibility for speeding up such iterative algorithm, like our proposed algorithm on calculating 3D time-space wave front propagation.

VII. ACKNOWLEDGMENTS

We would like to thank Atsushi Shigemura for developing the environment simulator component. This work is supported by NEDO (New Energy and Industrial Technology Development Organization, Japan) Intelligent RT Software Project.

REFERENCES

[1] M. S. Hassouna, A. E. Abdel-Hakim and A. A. Farag. "PDE-based robust robotic navigation". Image and Vision Computing, vol. 27, pp. 10-18, 2009.

[2] S. M. LaValle and J. Kuffner. "Rapidly-exploring random trees: Progress and prospects". In Proc. of Fourth Intl. Workshop on Algorithmic Foundations onRobotics (WAFR'00), 2000.

[3] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth". In Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS), 2003.

[4] S. M. LaValleand J. Kuffner. "Randomized kinodynamic planning". In Proc.IEEE Int. Conf. Robotics and Automation (ICRA), pages 473-479, 1999.

[5] S. Karaman and E. Frazzoli. "Incremental sampling-based optimal motion planning". In Robotics: Science and Systems (RSS), 2010.

[6] L. Jaillet, J. Cortes, and T. Simeon. "Sampling-Based Path Planning on Configuration-Space Costmaps". IEEE Transactions on Robotics,vol. 26, no. 4, pp. 635-646, Aug. 2010.

[7] M. Zucker, J. Kuffner, and J. A. Bagnell. "Adaptive workspace biasing for sampling-based planners". In Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), pp. 3757-3762 (2008).

[8] E. Plaku, L. E. Kavraki, and M. Y. Vardi. "A Motion Planner for a Hybrid Robotic System with Kinodynamic Constraints". In Proc. of the 2007 IEEE Int. Conf. on Robotics and Automation, pp. 692-697.

[9] A. Shigemura, Y. Ishikawa, J. Miura , and J. Satake. "People Movement Simulation in Public Space and Its Application to Robot Motion Planner Development". Proc. 2010 Int. Conf. on Advanced Mechatronics, pp. 504-509, Osaka, Japan, Oct. 2010.

[10] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W.K. Yoon. "RT-middleware: Distributed component middleware for RT (robot technology)". In: Proceedings of 2005 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 3555-3560, 2005.

[11] J. Sethian. "A fast marching level set method for monotonically advancing fronts". In Proc. Natl. Acad. Sci. volume 93, pages 1591-1595, 1996.

[12] H. Zhao. "A fast sweeping method for eikonal equations". Mathematics of Computation, 74:603-627, 2004

[13] I. Ardiyanto and J. Miura. "Heuristically Arrival Time Field-Biased (HeAT) Random Tree: An Online Path Planning Algorithm for Mobile Robot Considering Kinodynamic Constraints". In Proc. 2011 IEEE Int. Conf. on Robotics and Biomimetics, pp. 360-365, Phuket, Thailand, Dec. 2011.

[14] J. Won-Ki and W. Ross. "A Fast Iterative Method for Eikonal Equations". SIAM J. Sci. Comput. 30, pp. 2512-2534.

[15] V. Vonasek, J. Faigl, T. Krajnik, and L. Preucil. "A Sampling Scheme for Rapidly Exploring Random Trees using a Guiding Path". In Proc. 5th European Conf. on Mobile Robots, pp. 201-206, 2011.

[16] M. Zucker, J. Kuffner, and M. Branicky. "Multipartite RRTs for Rapid Replanning in Dynamic Environments". In Proc. IEEE ICRA 2007, pp. 1603-1609.

[17] K. Fujimura. "Time-Minimum Routes in Time-Dependent Networks". IEEE Trans. on Robotics and Automation, Vol. 11, No. 3, pp. 343351, 1995.