Proc. 9th Int. Conf. on Intelligent Autonomous Systems
pp. 85-93, Kashiwa, Japan, March 2006.

Mobile Robot Motion Planning Considering Path Ambiguity of Moving Obstacles

Hiroshi Koyasu and Jun Miura
Department of Mechanical Engineering, Osaka University
Suita, Osaka 565-0871, Japan
Email: {koyasu,jun}@cv.mech.eng.osaka-u.ac.jp

Abstract. This paper describes a motion planning method for mobile robot which considers the path ambiguity of moving obstacles. Each moving obstacle has a set of paths and their probabilities. The robot selects the motion which minimizes the expected time to reach its destination, by recursively predicting future states of each obstacle and then selecting the best motion for them. To calculate the motion for terminal nodes of the search tree, we use a randomized motion planner, which is an improved version of a previous method. Simulation results show the effectiveness of the proposed method.

Keywords. motion planning, dynamic environment, mobile robot

1. Introduction

Planning a path without collision with both static and dynamic obstacles is one of the fundamental function of mobile robots. Most of past research on motion planning in dynamic environments can be classified into two categories. One is reactive methods, which do not predict the paths of moving obstacles [5]. In such a reactive method, not an optimality but a safety is more important. The other is deliberative methods which are based on the assumption that paths of the moving obstacles are completely known; an optimal motion can be generated by employing a planning in space-time [3], or by using the concept of velocity obstacle [6,10]. However, it is difficult to predict paths of obstacles without uncertainty.

We divide the uncertainty of an obstacle motion into the path ambiguity (which path to choose) and the error in following a path. The latter can be handled by considering the range of uncertainty in motion planning [2]. A similar approach is, however, difficult to apply to the former because a very inefficient motion may be planned when the possible paths of an obstacle spread in a wide area. It is, therefore, necessary to cope with multiple path candidates.

Bennewitz et al. [1] proposed a motion planning method which considers multiple path candidates of obstacles. To avoid collision with an obstacle which has multiple path candidates, they use a heuristic cost function which is defined in space-time; the cost is set to be large near path candidates and proportional to the probability of each path candidate. Using this cost function, motion planning is done by using the A^* search

algorithm. The method, however, does not consider the change of the environment to be recognized by future observations.

Miura and Shirai [9] proposed to explicitly predict possible future states (i.e., which path a moving obstacle actually takes) of the environment and their probabilities for motion planning. By recursively searching for the optimal motion for each predicted state, they selected the optimal next motion minimizing the expected time to the destination. The method was, however, tested only for the case where there is only one moving obstacle with two path candidates and the possible motions of the robot are given in advance.

In this paper, we expand our previous method [9] so that it can handle multiple moving obstacles with multiple path candidates and plan robot motions according to the obstacle configuration and movements. We use a randomized motion planner, which is an improved version of a previous method. We show the effectiveness of the proposed method by simulation experiments.

2. Model of Moving Obstacles

In the environment, there are static and moving obstacles. Positions of static obstacles are given in advance. Each moving obstacle has its own set of candidate destinations and moves to one of them with avoiding collision with static obstacles. We represent a set of possible paths by using a tangent graph [8].

The initial probability of each obstacle taking one of its paths is given. The robot periodically observes the position and the velocity of each moving obstacle, and updates the probabilities of its path candidates. When a new observation is obtained, the probability of the jth candidate path of the ith obstacle is updated by

$$P(\mathit{path}_j^i \mid \boldsymbol{o}^i(t)) = \alpha P(\mathit{path}_j^i) P(\boldsymbol{o}^i(t) \mid \mathit{path}_j^i), \tag{1}$$

where $\boldsymbol{o}^i(t)$ is the observed position of the obstacle at time t, and α is a normalization constant. $P(\mathit{path}_j^i)$ is given by $P(\mathit{path}_j^i \mid \boldsymbol{o}^i(t-1))$. $P(\boldsymbol{o}^i(t) \mid \mathit{path}_j^i)$ is the likelihood of the path calculated by using the Gaussian positional uncertainty.

3. Predicting the State of Moving Obstacles

We define a state of a moving obstacle as a set of possible obstacle path candidates. The states can be classified into two cases. One is the case where only one candidate is remaining. We call this case a *fixed state*. The other is the case where multiple candidates are still remaining. We call this case an *ambiguous state*. An ambiguous state will be changed into a fixed state by future observations; the probability of a fixed state thus becomes large as time elapses.

3.1. Uncertainty of obstacle motion on a path

we assume that the error of a predicted position of an obstacle on a path is distributed within the so-called 3σ region of a 2D Gaussian. Its mean is the position predicted by assuming that the obstacle moves at the currently observed speed. The variance of the Gaussian perpendicular to the path is constant by supposing that an obstacle tries to follow the path as closely as possible. The variance along the path is proportional to a moving distance of the obstacle.

3.2. Calculating the probability of a state

We first consider the case where there is one obstacle with two path candidates. Fig. 1 shows the relationship between the prediction of the positional uncertainty and the path ambiguity in such a case. In the figure, path_1 and path_2 are drawn as black arrows. Black points are predicted positions on the path candidates at time t_1 and t_2. Let $R_i(t)$ be the

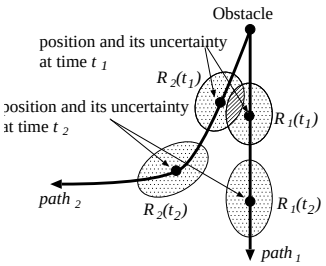

Figure 1. Prediction of the positional uncertainty and the path ambiguity of a moving obstacle.

3σ region of the Gaussian on $path_i$ at time t ($i = 1, 2$). If the obstacle is observed in the overlapped region of $R_1(t)$ and $R_2(t)$ (the hatched region in the figure), the state is an *ambiguous state*. The probability of this ambiguous state is calculated by

$$P(path_1)P(\boldsymbol{x}_1(t) \in R_2(t)) + P(path_2)P(\boldsymbol{x}_2(t) \in R_1(t)), \tag{2}$$

where $P(path_i)$ is the prior probability of $path_i$, and $\boldsymbol{x}_i(t)$ is the position at time t when the obstacle is actually on $path_i$. The first term of Eq. (2) is the probability that the obstacle on $path_1$ cannot be determined to be on either of the paths. $P(\boldsymbol{x}_1(t) \in R_2(t))$ is calculated by integrating the probability density function of $\boldsymbol{x}_1(t)$ in $R_2(t)$. The state is *fixed* if the obstacle is determined to be on either $path_1$ or $path_2$. The probability that the obstacle is on $path_1$ is given by There are two *fixed state*. One is the case where the obstacle is determined to be on $path_1$, and the other is the case of $path_2$. The probability of the case of $path_1$ is calculated by

$$P(path_1)P(\boldsymbol{x}_1(t) \notin R_2(t)). \tag{3}$$

Next, let us consider the case where there are n path candidates. The number of states of the obstacle is $2^n - 1$, that is, all combinations of paths. The probability of an ambiguous state where there are m remaining path candidates is calculated as follows. Let $L = \{l_1, \cdots, l_m\}$ be a set of indices of the remaining path candidates. For the case where the obstacle is on $path_{l_i}$, the probability that the m paths is still possible is equal to the probability that $\boldsymbol{x}_{l_i}$ is inside regions R_{l_j} ($j = 1, \cdots, i-1, i+1, \cdots, m$) and is outside the other $n - m$ regions. The weighted sum of such a probability in remaining path candidates becomes the probability of the ambiguous state with the above m remaining candidates as follows:

$$\sum_{i=1}^{m} P(path_{l_i})P(\boldsymbol{x}_{l_i}(t) \in \cap_{j \neq i} R_{l_j}(t), \boldsymbol{x}_{l_i}(t) \notin \cup_{k \notin L} R_k(t)). \tag{4}$$

Also, the probability of a fixed state where the obstacle is on $path_i$ is given by

$$P(path_i)P(\boldsymbol{x}_i(t) \notin \cup_{j \neq i} R_j(t)). \tag{5}$$

When there are multiple moving obstacles, a state of the environment is represented by a set of the states of all obstacles, and its probability is the product of the probabilities of the states of the obstacles. A state of the environment is called *fixed* if the paths of all obstacles are uniquely determined. Otherwise, a state is *ambiguous*.

4. Planning Method

The robot selects the motion which minimizes the expected cost. We use the time to reach the destination as the cost. We first predict possible future states of the environment and their probabilities. Then, by recursively searching for the optimal motion for each predicted state, we select the optimal next motion which minimizes the expected cost.

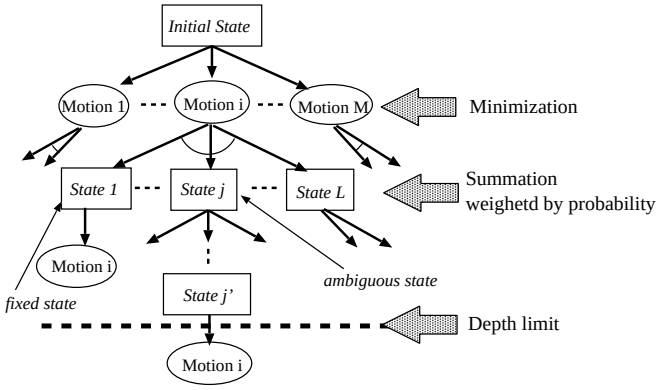

Figure 2. AND/OR tree search.

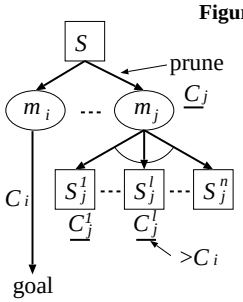

Figure 3. Prune a branch of the AND/OR tree.

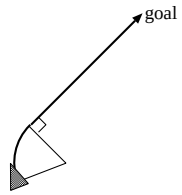

Figure 4. The path for calculating the lower bound.

This search process thus composes an AND/OR tree of robot motions and predicted states of the environment, as shown in Fig. 2.

Once the next motion is determined by this search process, the robot executes it and observes the environment to update the state. The robot plans the subsequent motion based on the updated state.

4.1. Search of the Motion Minimizing the Expected Cost

We explain the search of the motion minimizing the expectation of the cost using Fig. 2. We first pick a next motion and predict a set of states after the robot executes the motion. If a state in the set is *fixed*, we calculate the cost of the state by using the motion planner described in Sec. 4.3. If a state is *ambiguous*, we recursively search for the optimal motion (and thus the cost) for the state. We limit the search depth to avoid combinatorial explosion. When the search reaches the depth limit, we calculate a feasible robot motion which considers all possible obstacle paths, i.e., plan a robot motion by assuming that every remaining path candidate is actually taken by some obstacle. The expectation of the cost of the next motion is then obtained by calculating the sum of the expectation of the costs of all possible states weighted by their probabilities. We calculate the expectation of the cost for every candidate of the next motion, and select the one which has the minimum expectation of the cost.

4.2. Pruning of the AND/OR Tree by Using Lower Bound

To reduce the computation cost of the search, we perform a pruning of the AND/OR tree. Fig. 3 shows this pruning process, when searching for the motion minimizing the expected cost for state S.

Suppose that for motion m_i in state S, expected total cost C_i is calculated. This cost is the sum of the cost from the initial state to S and that from S to the goal state.

For another motion m_j, if its lower bound $\underline{C_j}$ is larger than C_i, m_j can be pruned. This lower bound changes as the total cost of a predicted state to be reached after executing m_j is obtained. Let S_j^k $(k = 1, \cdots, n)$ be the predicted states. After the costs C_j^k for S_j^k $(k=1, \cdots, l)$ are calculated, the lower bound is

$$\underline{C_j}^l = \underline{C_j} + \sum_{k=1}^{l} P(S_j^k) \left(C_j^k - \underline{C_j} \right). \tag{6}$$

If $\underline{C_j}^l > C_i$, we prune branch m_j without calculating the costs of remaining $n - l$ states.

Lower bound $\underline{C_j}$ is calculated as follows. We consider a motion pair which is composed of a turning motion and a straight motion to reach the goal from the current position, with neglecting any collisions with obstacles, as shown in Fig. 4. We examine the cost of such motion pair for a given set of turning radii and set the minimum cost to $\underline{C_j}$.

4.3. Calculating a Feasible Path

To calculate a feasible path for a fixed state (including the case where a robot has to make a plan by considering all possible path candidates at the depth limit), we use an improved version of the method proposed by Hsu et al. [4]. Their method is a randomized search method which considers non-holonomic constraints of the robot motion. The method picks a robot motion at random and generates a probabilistic roadmap of sampled state×time points, called milestones, connected by short admissible trajectories. The probabilistic roadmap consists of a tree of milestones whose root is the initial robot pose. This search process finishes by finding a milestone which is in the *endgame region*.

In the method, the completeness is an important issue, but the quality of planned path is not discussed. Since the method finishes the search by finding only one path, a found path may be much more inefficient than the optimal one. We improve their method in the following points:

- Give weights to milestones so that a wide space is explored.
- Iterate the path search until a sufficiently good path is found.

In this paper, the endgame region is defined as the set of robot poses from which a feasible path to the goal can be generated as the one shown in Fig. 4.

4.3.1. Weighting Milestones

In sampling milestones, if the weights of milestones are uniform, the tree may grow under a set of limited milestones; this may make a path quality low.

To deal with this problem, Hsu et al. [4] give a weight to a milestone inversely proportional to the number of other milestones which is in the neighborhood of the milestone in the configuration space. However, the distance between milestones under a specific milestone is not necessarily small when their depths are largely different. Rapidly-exploring Random Tree (RRT) [7], which uses the size of the Voronoi regions for weighting, may have the same drawback.

To make the tree grow relatively uniformly, we give the same weight to subtrees if their roots are at the same depth. Therefore the weight to milestone i with depth d_i and with b_i motions which have not been expanded is given by $(b_i/B)(1/B)^{d_i}$, where B is the number of robot motions (i.e., branching factor).

However, searching by using these weights is probabilistically equal to the breadth-first search; this may require much computation. So we put more weights on deeper milestones. The weight of the milestone is modified as $(b_i/B)(a/B)^{d_i}$, where $a\ (> 1)$ is a constant.

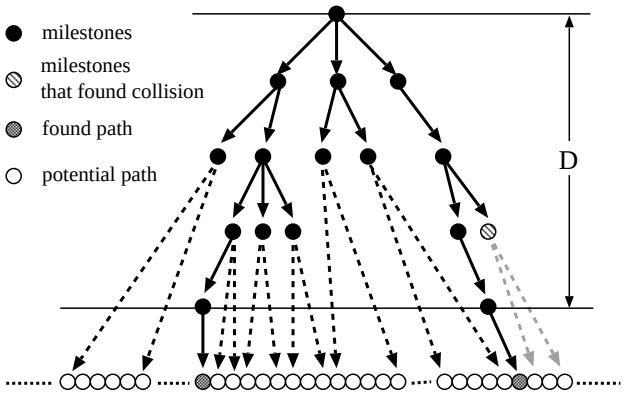

Figure 5. Number of potential paths on a milestone tree.

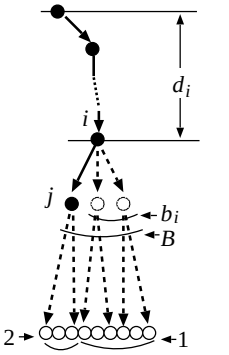

Figure 6. Milestones and their potential paths.

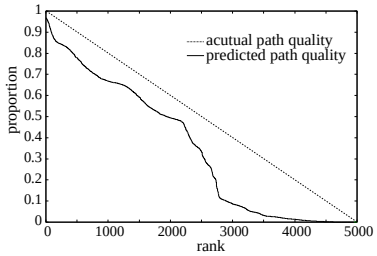

Figure 7. Predicted path qualities.

4.3.2. Evaluating the Quality of a Path

We iterate the path search several times and select the best path among the generated ones. We here consider the problem of determining when the path search is finished. To solve this problem, we need to be able to evaluate how good the current best path is. Therefore, we estimate the proportion of the number of paths whose costs are greater than that of the current best path to the total number of all possible paths. After a feasible path is found, if the lower bound of a milestone is larger than the cost of the path, no paths passing the milestone are more efficient than the feasible path. To estimate the proportion described above, we need to estimate the number of potential paths which passes a milestone.

We assume that milestones at the same depth have the same number of potential paths. Under this assumption, a deeper milestone has fewer potential paths; Fig. 5 illustrates this property. In the figure, D denotes the maximum depth of the tree, which is set to that of the current best path. The total number of the potential paths is B^D (B is the branching factor).

Fig. 6 illustrates how to calculate the number of potential paths of a milestone. In the figure, milestone i has a child milestone j. We consider that a potential path belongs to the deepest milestone on the path. So potential path set 1 belongs to milestone i and set 2 belongs to milestone j. A milestone which is a child of milestone i and has not been expanded has $B^{D-(d_i+1)}$ potential paths. If milestone i has b_i such children, its number of potential paths is $b_i B^{D-(d_i+1)}$.

If a milestone cannot be added to the tree due to collision, the potential paths passing it can never be obtained as feasible paths. When milestone j cannot be added, the number of potential paths passing it is B^{D-d_j}, and this number should be excluded from the total number of the potental paths.

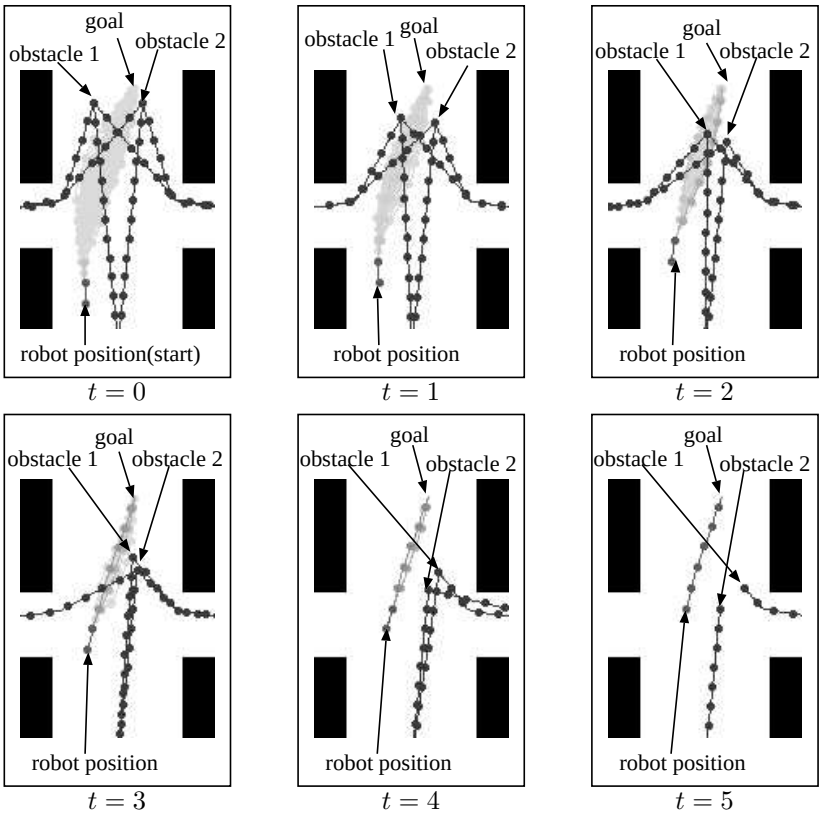

Figure 8. Simulation result.

Let I be the set of the indices of milestones whose lower bounds are greater than the cost of the current best path, and J be the set of the indices of milestones which cannot be added due to collision. The proportion of the number of potential paths whose costs are greater than that of the current best path to the total number of potential paths is thus estimated by

$$\frac{\sum_{i \in I} b_i B^{D-d_i-1}}{B^D - \sum_{j \in J} B^{D-d_j}}. \tag{7}$$

If the estimated proportion becomes greater than $1-\alpha$, the current best path is expected to belong to the best $100\alpha\%$ of the entire set of potential paths. If this condition is satisfied with a small α, we consider that a sufficient path is obtained. Currently, we use 0.1 as α.

To validate Eq. (7), we performed the following experiment. First, we generated 5000 paths from a milestone tree. Next, for each path, we calculated the estimated proportion given by Eq. (7) using the path as the current best path, and compared it with the actual proportion. Fig. 7 shows the comparison result. The estimated proportion is always less than the actual one mainly because the estimation is based on the lower bound. When the quality of a path is high enough, the estimated proportion is near to the actual; this suggests that this estimated proportion is a good measure of path quality.

5. Experimental Results

Fig. 8 shows the results of a simulation experiment. In the figure, white regions are free space, dark gray lines are path candidates of moving obstacles. Gray lines are planned

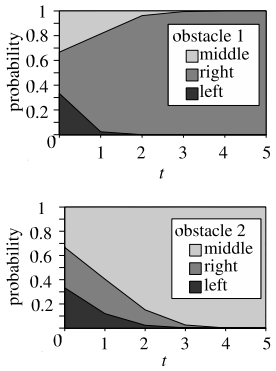

Figure 9. Probabilities of the paths of each obstacle.

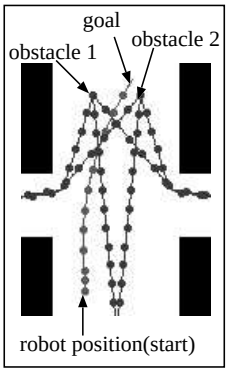

Figure 10. Result of motion planning without considering ambiguity.

robot paths, among which a darker line indicates a path with a higher probability. On paths of both the robot and the obstacles, circles are drawn with a fixed time interval.

In the experiment, the number of obstacles is two, the number of initial path candidates of each obstacle is three, and the depth limit of search is two. A motion of the robot is represented by a pair of the turning radius and the speed. The number of robot motions is nineteen. The prior probability of each path candidates is set equally.

Obstacle 1 actually takes the right path, and obstacle 2 takes the middle. Fig. 9 shows the change of the probabilities of paths for each obstacle. For obstacle 1, the probability of taking the right path becomes dominant at $t = 1$, but the ambiguity of paths remains until $t = 4$. For obstacle 2, the probability of taking the middle path becomes dominant at $t = 1$, but the ambiguity of paths remains until $t = 4$.

At time $t = 0$, since all paths of the obstacles are equally probable, the robot selects the motion to move a position where the robot can safely avoid collision with the obstacles even if they will take any paths. At time $t = 1$, the situation is still highly ambiguous and the robot selects a motion similar to the one selected at time $t = 0$. At time $t = 2$, since the path of each obstacle is almost uniquely determined, only a few robot motions in a generated plan have high probabilities. At time $t = 3$, the robot now knows that both obstacles will not take their left paths and generates a plan in which the probability of robot motions which will cross over the left paths. At time $t = 4$, there is no obstacle in front of the robot. However, since the environment is not fixed, the robot generates two slightly different paths due to the randomness of the planner. At time $t = 5$, the paths of the obstacles are fixed, and the robot plans a single path towards the destination.

The computation time at $t = 0$ is 140 [sec] using an Athlon 2200+ PC. At $t = 1$, the time is 9.5 [sec]. After $t = 1$, the time does not exceed 0.5 [sec]. The computation time is decreased with time elapses, because the ambiguity of the environment is decreased.

We also compared our method with the method which plans a robot motion to avoid all possible path candidates of obstacles. Fig. 10 shows the planning result at $t = 0$ by this method in the same situation as the one shown in Fig. 8. In the plan, the robot moves slowly until $t = 2$ to wait for the obstacles to go away. The expected cost of the plan is 12.90 [sec]. On the other hand, our method selects a faster motion in the same situation, and the expected cost is 10.99 [sec]. This result shows the effectiveness of the proposed method.

6. Conclusion

This paper has described a path planning method which considers path ambiguity of moving obstacles. The method predicts possible future states and their probabilities, and searches for a robot motion for each predicted state. This prediction and search are recursively performed until every state becomes *terminal* (i.e., the ambiguity of a state is resolved or the search reaches the depth limit). The method finally selects the next motion which minimizes the expected time to the destination.

The proposed method can deal with fairly general motion planning problems with arbitrary static obstacle configurations and multiple moving obstacles with multiple path candidates. The path planner which we use for calculating the costs of terminal nodes is an improved version of a previous randomized method; it iteratively generates a set of paths until a sufficiently good path is generated. The number of iteration is determined based on the quality evaluation of the current best path.

The current method still needs too much calculation time to be used on-line. A future work is to consider the trade-off between the quality of a planned path and the computational cost in order to appropriately control the planning time.

References

[1] M. Bennewitz, W. Burgard, and S. Thrun. Adapting navigation strategies using motions patterns of people. In *Proc. of the 2003 IEEE Int. Conf. on Robotics and Automation*, pages 2000–2005, 2003.

[2] A. Elnagar and A. Basu. Local Path Planning in Dynamic Environments with Uncertainty. In *Proc. of IEEE Int. Conf. on Multisencor Fusion and Integration for Intelligent Systems*, pages 183–190, 1994.

[3] K. Fujimura. Time-Minimum Routes in Time-Dependent Networks. *IEEE Tran. of Robotics and Automation*, 11(3):343–351, 1995.

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

[5] S. Ishikawa. A method of indoor mobile robot navigation by using fuzzy control. In *Proc. 1991 IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, pages 1013–1018, 1991.

[6] F. Large, S. Sekhavat, Z. Shiller, and C. Laugier. Towards real-time global motion planning in a dynamic environment using the nlvo concept. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2002.

[7] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, Wellesley, MA, 2001.

[8] Y.-H. Liu and S. Arimoto. Path planning using a tangent graph for mobile robots among polygonal and curved objects. *Int. J. of Robotics Research*, 11(4):376–382, 1992.

[9] J. Miura and Y. Shirai. Probabilistic uncertainty modeling of obstacle motion for robot motion planning. *J. of Robotics and Mechatronics*, 14(4):349–356, 2002.

[10] Z. Qu, J. Wang, and C. E. Plaisted. A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles. *IEEE Transactions on Robotics*, 20(6):978–993, 2004.