A Method of Planning Movement and Observation for a Mobile Robot Considering Uncertainties of Movement, Visual Sensing, and a Map*

Toshihiko KANBARA**, Jun MIURA***,
Yoshiaki SHIRAI***, Akira HAYASHI**
and Shigang LI**

We present a method of route selection for a vision guided mobile robot with an odometry system, given an uncertain map. The robot first generates candidates of the routes, which are expanded to possible loci in consideration of the map uncertainty. It then makes an observation and movement plan along each locus and computes the minimum traveling time. Uncertainties of movement and visual sensing are considered in this step. Finally, for each candidate route, the robot computes its expected traveling time from probabilities of loci. It selects the route with the least expected traveling time as the optimal route. The advantage of the planning method is that the observation and movement planning are incorporated in path planning to obtain the optimal plan. The three sources of uncertainty are considered in a unified manner in our method. The method has been implemented and tested to validate our claim.

Key Words: Computer Control, Moving Robot, Image Processing, Reasoning, Path Planning, Observation Planning, Stochastic Map, Uncertainty

1. Introduction

In recent days, many autonomous robots have been developed. Most of them are equipped with internal and external sensors. One task of such a robot is to reach a destination accurately or quickly without collision. Given the task, it first refers to a given map and generates a plan for sensing and for moving. Then, it executes the plan.

Such a robot, however, cannot accomplish his purpose perfectly. Main reason for this difficulty is the *uncertainties* involved. The localization data of dead reckoning includes an error (uncertainty of

movement). This error accumulates as robot moves farther.

Although this error can be reduced by external sensors, the external sensors also have measurement error (uncertainty of observation). In addition, the given map may have error (uncertainty of the map). In this paper, we propose a new method to plan the robot where to move, where to observe, and what to observe for reaching the goal certainly in minimum time, considering the three sources of uncertainties.

Many researchers^{(1),(2)} assume that there is no uncertainty and propose methods to make a plan for the shortest route. Rosa et al.⁽³⁾ consider the uncertainty of movement, but don't consider the uncertainty of observation. We consider both of the uncertainties.

Many methods for determining features to be observed on the given route are proposed. Curran et al.⁽⁴⁾ and Betke et al.⁽⁵⁾ propose planning methods for reducing the uncertainties of localization by Kalman filter. But they don't select proper features. Only the nearest features are used for observation. Komoriya et al.⁽⁶⁾ select the proper features. In the above three methods, the robot moves along a given route.

* Received 18th October, 1999, Japanese original: Trans. Jpn. Soc. Mech. Eng., Vol. 65, No. 629, C (1998), p. 130-137 (Received 23rd March, 1998)

** Department of Information Machines and Interfaces, Hiroshima City University, 3-4-1 Ozuka-higashi, Asa-minami-ku, Hiroshima 731-3194, Japan. E-mail: kanbara@im.hiroshima-cu.ac.jp

*** Department of Mechanical Engineering for Computer-Controlled Machinery, Osaka University, 2-1 Yamadaoka, Suita, Osaka 565-0871, Japan

However, our robot not only selects proper features to be observed, but also plans proper route.

In our planning method, the robot first generates multiple candidates of *routes* from an uncertain map and a goal. The route has a topological and qualitative description. It is composed of free space blocks. Second, the robot expands each route into possible *loci*. The locus has a quantitative description. It is composed of metric curves. From uncertainty of the map, the probability is assigned to each locus. For each locus, the robot generates an optimal plan of observations and movements so that the robot can reach the goal in minimum time. Finally, the robot computes the expected traveling time for all routes and selects an optimal route.

Section 2 describes uncertainties of the robot and the map. Section 3 shows how to generate the route and the locus. Section 4 shows our method for planning observations and movements. Section 5 shows computation of the expected time and selection of the optimal route. Section 6 shows the result of simulation. Section 7 gives the conclusion.

2. Uncertainties of the Robot and the Map

2.1 Uncertainties of the robot

We approximate the robot's shape by an enclosing cylinder. Its radius is denoted by r_R . The position and orientation of the robot is represented by a state vector: $S_R = [X_R, Y_R, \theta_R]^T$. The probability density function $p_R(S_R)$, which expresses the uncertainty of the robot, is generated from a three dimensional gaussian distribution. The covariance matrix (3×3) of the gaussian distribution is denoted as A_R .

2.2 Visual feedback control

Using dead reckoning and visual sensing, the robot can localize himself without stopping. Generally, the processing time for image t_p , is longer than the measurement time for dead reckoning, t_c . Therefore, the robot can measure repeatedly while one image is processed. At every t_c time, the robot controls the steering so that the robot moves along a given locus.

The uncertainty of the robot's position is indicated by an ellipse, as shown in Fig. 1. This ellipse is

Fig. 1 Updating the error ellipse after movement and observation

called as *error ellipse*. It is generated from the marginal distribution of $p_R(S_R)$, and means the boundary of the region of uncertainty. The size of ellipse increases during the movement, and can be reduced by an observation (i.e. image processing).

In Fig. 1, the robot is at position A (at the center of the ellipse e_1) with rightward orientation. The robot begins to move by using dead reckoning and processes the image which has been taken at position A. When the robot reaches position B, the result of processing becomes available. Using the result, the uncertainty can be reduced from e_1 to e_3 . By fusing e_3 and the recorded sequence of internal sensor values between A and B, a new error ellipse e_4 is estimated. Then, the robot update the current uncertainty e_2 with e_4 . This cycle of sensing and control is iterated. We call this cycle as an *image-processing cycle*. This online estimation method is known as the non-stop navigation^{(8),(9)}.

We assume that the time t_p is constant in spite of a possible change in the image complexity. Thus, the moving distance in one image-processing cycle depends only on the moving velocity V . We set the upper limit, V_{max} , on the velocity so that the uncertainty of the orientation cannot become too large.

2.3 Uncertainty of the map

The map describes the placements (locations and orientations) of all objects. We approximate that all objects are polygonal pillars. The number of all objects is denoted by n_o . We denote the location and orientation of j th object as (X_j, Y_j, θ_j) .

The state M of the map is a $3n_o$ dimensional vector $M = [X_1, Y_1, \theta_1, \dots, X_{n_o}, Y_{n_o}, \theta_{n_o}]^T$. The uncertainty of the map is given by probability density function $p(M)$. We assume $p(M)$ is Gaussian. We associate the distribution with the so called *error ellipsoid* whose radius along each main dimensional axis is three times of the standard deviation. We define the inside of the ellipsoid as the *region of uncertainty* of the map. In this region, the gaussian is normalized into $p(M)$.

3. Generating the Routes and the Loci

3.1 How to generate the routes

The robot generates many candidate routes bound to goal as follow. Here, we assume that all objects are placed at the average state $\bar{M}$. First, many *elements* of the routes are generated by using a Voronoi diagram⁽⁷⁾ such as Fig. 2(a). This Voronoi diagram is composed of lines and parabolas. In Fig. 2, all elements are denoted by, $E_1, E_2, \dots$, and E_{24} . Each element is generated from two original objects. For example, parabola E_8 is generated from the left wall and the upper pillar. *Segments* are generated by

Fig. 2 Method of generating routes

merging elements between the same object pair. In Fig. 2(b), a set of elements which are enclosed by dashed line is the segment. For example, elements E_8 and E_{10} are merged into segment S_4 .

If the minimum distance from one segment to the nearest object is larger than the size of the robot, the robot can move along the segment without collision. We define such a segment as a *traversable segment*.

We represent all traversable segments by nodes, then connect the adjacent segments by edges to generate a graph. The start node (S_7) and the goal node (S_5) are the segments which are nearest to the start point and the goal point respectively. Using this graph, the problem of generating routes is reduced to the graph search problem. Then, the robot searches the graph for all paths between the start node and the goal node.

Since the map is uncertain, the robot examines whether each path is traversable or not, when the placement M changes within the region of uncertainty (see Section 2.3). All paths which consist of traversable elements are selected as candidates of the routes, $R^{(i)} (i=1, 2, \dots, n_r)$.

3.2 Definition of the safety-width

Even if the robot moves at its maximum speed $V_{\max}$, it can avoid collision by keeping enough distance from both the left and right side objects. We call this distance *safety-width* and this length is predetermined as follows.

Since the robot navigates in a known cluttered indoor environment, we assume that the robot can find at least one visible feature within a certain distance, $L_{\max}$. Let us consider a scenario in which the robot moves along a straight locus at its maximum speed, $V_{\max}$, and observes a vertical edge at the distance of $L_{\max}$ in every image-processing cycle (see Fig. 3). Then, the maximum uncertainty in the direction perpendicular to the locus becomes constant, H . When the robot returns to the locus, the robot must move on an arc trajectory because of the non-holonomic constraints, and this movement causes a small additional

Fig. 3 Computation of the safety-width $H_{\max}$

Fig. 4 An example of expanded regions and generated locus

displacement away from the locus. Thus, we compute the safety-width $H_{\max}$ by adding the displacement to H .

We expand the objects by ' r_r ' (defined at Section 2.1), and treat the robot as a point. The light gray regions in Fig. 4 are generated by expanding the objects. Then the robot can detect (possible) collisions by finding overlaps between the expanded object region and the error ellipse.

3.3 How to generate the loci

For each route, if the placement of objects changes within the uncertainty region, then the corresponding locus also changes. Therefore, the robot generates multiple loci for each placement by the following procedures, in which the notion of the safety-width is used.

(1) For each placement of the objects, we build a Voronoi diagram as explained in the route generation phase (see Section 3.1). According to the property of Voronoi diagrams, the distances from one point on a line or parabola element to the nearest two (or more) objects are equal. If there is a point on the element with the distance which is smaller than $[H_{\max} + r_r]$, it is dangerous to move along the element at $V_{\max}$ speed. We call this dangerous part of the segment a *dangerous zone*. These dangerous zones are determined as parts of the locus in the first place.

(2) For safely movement at $V_{\max}$ speed, the robot must not enter regions drawn by dashed lines in Fig. 4. These regions are generated by expanding objects with the $[H_{\max} + r_r]$. Under this constraint, the robot generates part of the locus so that the length of the locus is minimized. We call the part of the generated locus in this *free space zone*. In this zone, the robot

can move at $V_{\max}$ speed without collisions.

In Fig. 4, A, B, and C parts are generated by step (1), and D and E parts are generated by step (2) respectively.

4. Planning of Observations and Movements

4.1 Strategies for planning

To minimize the total traveling time, we adopt a strategy that the robot maximizes the moving distance in each image-processing cycle. For avoiding collisions, it is desirable that the robot minimizes its current uncertainty. We also note that the result of observation becomes available at next image-processing cycle. By considering all these factors, we propose an algorithm of offline planning observations and movements as follows.

(1) After the processing of the observation in the previous cycle is finished, the robot, using the result, reduces the uncertainty at the beginning of the previous cycle. By adding the reduced previous uncertainty to the uncertainty of movement which is caused by moving the planned moving distance for the previous cycle, the robot updates the current uncertainty.

(2) The robot plans for the optimal object to be observed so that the robot can minimize the uncertainty on the basis of the current uncertainty which is computed at step (1).

(3) Similarly, on the basis of the current uncertainty, the robot plans for the longest moving distance so that the robot can not only avoid collision at the current cycle, but can also return to the locus in the next cycle from anywhere inside the current uncertainty. If the above two conditions are satisfied, we define that the robot is *recoverable*.

Repeating the above steps, the robot plans a sequence of the observations and the movements along the locus. The result of the observation depends on the actual location inside the uncertainty. However, since this planning is offline and this location cannot be known a priori, we suppose that the robot will get the result of observation from the center of the error ellipse.

4.2 Planning observations and movements

We explain our planning method in more detail, according to whether the robot is in a free space zone or in a dangerous zone.

4.2.1 Selection of edges to be observed in the free space zone Since the robot need not slow down the speed in the free space zone, the moving distance can be determined to be the longest length. For each cycle, the robot observes one vertical edge in this zone. For each vertical edge visible from the current position by panning the camera head within 90° , the robot computes the possible reduction in the

uncertainty, if the observation is made. Then, the robot selects the best vertical edge which leads to the maximum reduction in the uncertainty (i.e. the maximum reduction in the area of the error ellipse).

As a concrete example, see Fig. 5(a). There are four objects. Nine visible edges are picked up, and then the reduced uncertainties are computed. In Fig. 5(b), each inner ellipse shows the reduced uncertainty by observing the corresponding edge, while each outer ellipse shows the original uncertainty before the observation.

Although the nearer edge can reduce robots' positional uncertainty more drastic in the many cases, our robot computes the reduced uncertainty for an original correlation between the orientational uncertainty and the positional uncertainty. An example of the correlation is shown in Fig. 6. The robot located in B position where is the center of the error ellipse. Three arrows mean centers of probability distribution of robots' orientation in three position A, B and C. For the correlation, these arrows are not parallel. When the robot observes edge D and E, the robot gets an observation angle θ in B position. Although the robot can get θ in both two position A and C from edge D, the robot can get θ in neither A nor C from edge E, considering the orientational uncertainties (light gray

Fig. 5 Edge selection for observation in a free space zone

Fig. 6 A case of little uncertainty reduction by observing a near edge

(a) Placement of edges (b) Result of observation
Fig. 7 Edge pair selection for observation in a dangerous zone

(a) Edge 1 and 3 (b) Edge 1 and 4
Fig. 8 Edge pairs for observation

sector regions). Thus, the probability which the robot locates in A and C can be reduced by father edge E rather than nearer edge D.

4.2.2 Selection of edges to be observed in a dangerous zone In a dangerous zone, the robot may collide with one of two nearest objects if it moves at its maximum speed, $V_{\max}$. To avoid collisions, the robot moves slowly and observes a pair of edges of these nearest objects, one edge is horizontal and the other is vertical. Of all edge pairs, the robot selects the best pair which leads to the minimum uncertainty after the observation is made.

Figure 7(a) shows the robot in a dangerous zone. The visible vertical edges (No. 1 or 3) are indicated by bold black points, and the visible horizontal edges (No. 2 or 4) are indicated by thick lines. In this example, four candidate pairs are picked up. In Figs. 8(a) and (b), two possible camera views which the robot takes by observing the pair of No. 1 and 3, and the pair of No. 1 and 4, are shown. The optimal pair (No. 1 and 4) is selected.

4.3 Determination of moving distance

The robot needs to compute the bound on the moving distance in order to keep the robot to be recoverable (defined at Section 4.1). However, it is difficult to examine all cases associated with starting from somewhere within the current error ellipse and moving for certain distance. Therefore, to determine whether the robot is recoverable or not, we examine only four points as candidates of one most critical point. These points exist where the two main axes cross the ellipse boundary.

Fig. 9 Recovery from the most critical location

In Fig. 9, the robot determines whether moving to be e_1 position will keep himself recoverable. Of the four points examined, the one with a black bold point on the e_1 ellipse is found to be the most critical point. At this point, the orientation with the highest probability (i.e. the center of the gaussian distribution) is indicated by the thick line arrow. The robot generates the trajectory (the dashed arrow) for returning to the locus, then examines whether the robot is recoverable or not through the trajectory (actually not recoverable). Although the robot cannot move to e_1 position, it can move to e_2 position.

4.4 Computation of total traveling time

For each of the candidate routes $R^{(i)}$ ($i=1, 2, \dots, n_r$), the robot can generate the corresponding loci from the placement $\mathbf{M}$ of the objects. By the method explained above, the best sequence of movements and observations is planned along each locus. From the sequence, the total number $q^{(i)}$ of image-processing cycles is computed. The moving time $t_j^{(i)}(\mathbf{M})$ in the final cycle is also computed. From these values, the total traveling time $T^{(i)}(\mathbf{M})$ along the locus for the placement is estimated by

$$T^{(i)}(\mathbf{M}) = t_p \cdot (q^{(i)} - 1) + t_j^{(i)}(\mathbf{M}), \quad (1)$$

where t_p is time for image processing per cycle (see Section 2.2).

5. Route Selection

5.1 Computation of the expected time

By changing placement $\mathbf{M}$ within the region of uncertainty (see Section 2.3), the corresponding locus of route $R^{(i)}$ is also changed. All the objects which have caused the change of the locus are collected to form a set. The number of elements of the set is denoted by $n_i (\leq n_o)$. The partial state vector $\mathbf{M}^{(i)}$ is represented by $3n_i$ variables, and its probability function $p(\mathbf{M}^{(i)})$ is obtained as the marginal distribution of the $p(\mathbf{M})$. From these functions, the expected traveling time is computed by the following equation:

$$E[T^{(i)}] = \int T^{(i)}(\mathbf{M}^{(i)}) \cdot p(\mathbf{M}^{(i)}) d\mathbf{M}^{(i)}. \quad (2)$$

Function $p_d(\mathbf{M}^{(i)})$ is generated by discretizing

Fig. 10 A problem to reach a goal through a gap

Fig. 11 Traveling time

$p(\mathbf{M}^{(i)})$. By discretizing $\mathbf{M}$ into finite s placements $\mathbf{M}_j^{(i)} (j=1, 2, \dots, s)$ and assigning to each of them the probability, the expected time can be computed. The total number of discretized placements, s , is determined by considering the tradeoff between the amount of computation and the required accuracy. By the discretizing, the expected traveling time is approximated by the following equation:

$$E[T^{(i)}] \approx \sum_{j=1}^s T^{(i)}(\mathbf{M}_j^{(i)}) \cdot p_d(\mathbf{M}_j^{(i)}). \quad (3)$$

5.2 Justification for using the expected time

We explain, by an example problem, why our method uses an expected traveling time rather than a traveling time in one expected placement.

In Fig. 10, the robot moves forward, passes through a gap between two objects and reaches the goal. The total traveling time depends on the size of the gap d in the figure. If d is small, many rear edges are occluded by the two objects in front and the uncertainty of the robot cannot be reduced enough. As a result, the total time is long. Otherwise, the robot can reduce the uncertainty. Therefore, the total time is shortened drastically as shown in Fig. 11.

In Fig. 12, two expected times are shown. The thick curve shows the expected time with larger uncertainty in the gap size, and the dashed curve shows one with smaller uncertainty. For the computation, two gaussian probability density functions $p(d)$ are generated with the same mean value but with different variances.

These two curves cross each other. In the left side of the cross point, the expected time with small gap uncertainty is shorter than that with large gap

Fig. 12 Expected traveling time

uncertainty. In the right side, the relation between two expected times is reversed. Therefore, if there are two gaps with the same mean value but with different variances, we need to compute their expected times in order to determine which gap is better in terms of the expected traveling time.

5.3 Reducing amount of computation by setting subgoals

For computing Eq. (3), the set of state vectors $\mathbf{M}^{(i)}$, must be determined properly. To see this, suppose we discretize each dimension of the state vector into n_u values. Since each state vector has $3n_i$ dimensions, $s = n_u^{3n_i}$ holds, and we can see that s becomes exponentially large as the number n_i grows.

For reducing the amount of computation, we set some subgoals on each route. The subgoals are put in free space zones so that each of the dangerous zones are divided by the subgoals. For example, when there is one free space zone between two dangerous zones, a subgoal is generated in the center of the two dangerous zones. Using subgoals, a route is divided into some parts so that one part has only one dangerous zone certainly. We now explain how much reduction in the amount of computation is obtained by setting subgoals.

The total number of divided parts is denoted by f . The partial traveling time which the robot spends for moving from the $(l-1)$ th subgoal to the l th subgoal is denoted by $T_l^{(i)}(\mathbf{M}^{(i)})$.

When the subgoals are set, the robot uses the partial state vector $\mathbf{M}^{(i,l)}$ which contains only the states of m_l objects that are related with the changes of locus in the $(l-1)$ th part of the route. This vector is discretized into s_l states. This s_l could be set as $s_l = n_u^{3m_l}$ as explained in the first paragraph of the section. The probability density function $p_d(\mathbf{M}_k^{(i,l)}) \{k=1, 2, \dots, s_l\}$ is computed by discretizing the marginal distribution $p(\mathbf{M}^{(i,l)})$. From these values, the expected time $E[T^{(i)}]$ is computed by Eq. (4).

$$E[T^{(i)}] = \sum_{l=1}^f \left(\sum_{k=1}^{s_l} T_l^{(i)}(\mathbf{M}_k^{(i,l)}) \cdot p_d(\mathbf{M}_k^{(i,l)}) \right) \quad (4)$$

In this computation, the amount is reduced from

$n_u^{3n_i}$ to $\sum_{l=1}^f n_u^{3m_l}$. Since $n_i = \sum_{l=1}^f m_l$ and $f > 2$ hold, the amount of computation is reduced certainly, i.e. the next inequality holds:

$$n_u^{3n_i} = n_u^{(\sum_{l=1}^f 3m_l)} = \prod_{l=1}^f n_u^{3m_l} \gg \sum_{l=1}^f n_u^{3m_l}$$

After computing the current partial time $T_l^{(i)}(\mathbf{M}_k^{(i,l)})$, the robot computes the next state $S_{R(l)}$ and the average of the uncertainty of $\Delta_{R(l)}$ for computing next partial time $T_{l+1}^{(i)}(\mathbf{M}_k^{(i,l+1)})$.

6. Simulation

For verifying the effectiveness of our method, we ran our planner in a simulated environment. The planner generates a sequence of observations and movements for reaching the goal in minimum expected time in an environment shown in Fig. 13.

We set the world coordinate system, $X-Y$, whose origin is at the bottom left corner of the figure and whose X axis is in the horizontal direction. The start and goal positions are given by their coordinate values. There is a wide free space in the middle of the figure. From the start to this free space, there are three routes: A, B, and C. From the free space to the goal, there are also three routes: D, E, and F. We set a common subgoal point of route A, B, and C. This common subgoal is determined manually in this simulation. The robot selects the best route from these nine route combinations ($A \rightarrow D$, $A \rightarrow E$, ..., $C \rightarrow F$).

The orientational uncertainty of all objects is zero. The two positional (x and y) uncertainties are independent and have the same standard deviation. The standard deviations for Object 1 and Object 2 are 50/3 [mm] and 100/3 [mm].

For Object 1 and Object 2, their probability density functions are discretized to 9 and 25 states, respectively. Therefore, the computation of the expected time for the route A (Start $\rightarrow$ A $\rightarrow$ Subgoal) requires 9 states. For the route B and route C, it requires 225 (9×25) and 25 states, respectively. The result of the computation of the expected time is shown in the left side of Table 1.

The numbers of states of the route D, E, and F are 9, 225 and 25 respectively. The results of expected

time which are computed for the route D, E and F, are shown in the right side of Table 1. From these results, the robot selects the best route: $B \rightarrow E$. The sequence of planned observations and planned movements is shown in Figs. 14, 15. The center point of each error ellipse means the position where the robot observes edges. The numbers for the observed edges indicate the points where the observations are made.

While the robot has given initial uncertainty at the start point in Fig. 14, we suppose that the robot has an average uncertainty of the subgoal as initial uncertainty in Fig. 15. Without the subgoal, the robot would require $81 + 2 \cdot 225 + 625 + 2 \cdot 2025 + 2 \cdot 5625 + 50625 = 67081$ times of computation. By our method, however, this amount of computation is reduced to $(9 + 25 + 225) \cdot (1 + 3) = 1036$ times.

In the simulation, we modeled the robot movement and the image processing as follows. The robot has three wheels, and the front wheel is driven and steered. Odometries are attached to the front wheel. The measurement time t_c is 0.075 [sec]. The size of

Table 1 Expected time to/from subgoal [sec]

	time		D	E	F
A	23.16	time after A	22.04	20.17	22.60
B	20.22	time after B	21.01	20.13	23.13
C	21.03	time after C	20.23	21.74	23.04

Fig. 14 Planning observations and movements (B)

Fig. 15 Planning observations and movements (E)

Fig. 13 Routes from start and routes from subgoal

camera view is $640 [\text{pixel}] \times 480 [\text{pixel}]$, and the focal length is $400 [\text{pixel}]$. We use the distance $L_{\max}$ ($2900 [\text{mm}]$) and the speed $V_{\max}$ ($500 [\text{mm/sec}]$). From the processing time t_p ($3 [\text{sec}]$), we compute the safety width as $322 [\text{mm}]$. This t_c is determined from the recent research of Moon et al.⁽¹⁰⁾

7. Conclusion

We have proposed a new method of planning the sequence of observations and movements considering the uncertainties of movement, observation, and the map. The system works in three steps: (1) The robot generates candidate routes, and expands each route to loci according to the placement of objects. (2) On each locus, the robot plans the optimal sequence of observations and movements, and then computes the total traveling time. (3) From traveling time about all loci on each route, the robot computes the expected traveling time on the route. From all the routes, the robot selects the route with the minimum expected time.

Acknowledgement

A part of this research was supported by Grant-in-Aid for Scientific Research on Priority Areas "Intelligent Robot", from the Ministry of Education, Science and Culture.

References

- (1) Ogata, H., Arai, T. and Ota, J., Trajectory Planning for Mobile Robots in Time-Varying Environments Considering User's Specification, *Journal of Robotics Society of Japan*, (in Japanese), Vol. 12, No. 6 (1994), p. 905-910.
- (2) Namgung, I. and Duffy, J., Two Dimensional Collision-Free Path Planning Using Linear Parametric Curve, *Journal of Robotic Systems*, Vol. 14, No. 9 (1997), p. 659-674.
- (3) De la Rosa, F., Laugier, C. and Najera, J., Robust Path Planning in the Plane, *IEEE Trans. on Robotics and Automation*, Vol. 12, No. 2 (1996), p. 347-352.
- (4) Curran, A. and Kyriakopoulos, K.J., Sensor-Based Self-Localization for Wheeled Mobile Robots, *Journal of Robotic Systems*, Vol. 12, No. 3 (1995), p. 163-176.
- (5) Betke, M. and Gurrts, L., Mobile Robot Localization Using Landmarks, *IEEE Trans. on Robotics and Automation*, Vol. 13, No. 2 (1997), p. 251-263.
- (6) Komoriya, K., Oyama, E. and Tani, K., Planning of Landmark Measurement for the Navigation of a Mobile Robot, *Journal of Robotics Society of Japan*, (in Japanese), Vol. 11, No. 4 (1993), p. 533-540.
- (7) Latombe, J.-C., *Robot Motion Planning*, (1991), Kluwer Dordrecht.
- (8) Maeyama, S., Ohya, A. and Yuta, S., Non-Stop Outdoor Navigation of a Mobile Robot, *Proc. of Int. Conf. on IROS '95*, Vol. 1 (1995), p. 130-135.
- (9) Kosaka, A., Meng, M. and Kak, A.C., Vision-Guided Mobile Robot Navigation Using Retroactive Updating of Position Uncertainty, *Proc. of Robotics and Automation*, Vol. 2 (1993), p. 1-7.
- (10) Moon, I.H., Miura, J. and Shirai, Y., Dynamic Motion Planning for Efficient Visual Navigation under Uncertainty, *Proc. the 5th Symposium on Intelligent Autonomous Systems*, (1998), p. 172-179.