Algorithm Of A Manipulator Movement Amidst Unknown Obstacles

P.K.Lopatin

Computer Science Department
Siberian Aerospace Academy
31, Krasnoyarskiy Rabochiy av., 660014, Krasnoyarsk, RUSSIA, P.O.Box 486
plopatin@aport.ru

Abstract

The algorithm for a manipulator movement amidst unknown obstacles is presented. The algorithm guarantees the reaching of a target position in finite number of steps. Manipulator may have arbitrary but finite number of links. The obstacles may have arbitrary disposition, shapes and dimensions. Their number may be arbitrary. It is supposed that the obstacles are stationary and it is supposed that the target position is reachable. Also is given the simplified algorithm and the results of the computer imitation of the manipulator movement based on the simplified algorithm.

1. Introduction

In order to increase the efficiency and productivity of fabrication robots and manipulators are widely used in different spheres of industry. The robot must be as autonomous as possible and it must effectively operate in natural environment.

In the beginning of robotic era robots operated in workspace which was free of obstacles. Later the works dedicated to the invention of algorithms for the control of manipulators in the presence of obstacles began to appear. There are algorithms which guarantee finding a trajectory in the presence of known obstacles if such trajectory exists [4, 5, 6]. Some authors use the artificial potential methods (see, for example, [2]). In this method robot is represented by a point, regions in the workspace that are to be avoided are modelled by repulsive potentials and the region to which the robot is to move is modelled by an attractive potential. In general situation there is no guarantee that a collision-free path will always be found if one exists [1]. There are different heuristical searching methods (for example, A* class of searching methods, see for example, [12]) which find the trajectory avoiding obstacles (even an optimal one), if such trajectory exists. It is possible to use such methods in case we have full information about free and forbidden points before the beginning of the movement. Then a powerful computer may calculate a preliminary trajectory and after that the manipulator may realize this trajectory. But in case of unknown obstacles the manipulator has to investigate its workspace and plan its trajectory simultaneously. Then such a difficulty arises that graph algorithms of a route searching demand the breadth searching, otherwise the reaching of the target configuration will not be guaranteed. But during the breadth searching it is necessary to switch from one node **q** to another node **q*** which may be not adjacent. Then the problem of manipulator moving from **q** to **q*** arises and as a result the total sum of the manipulator's movements becomes very big [8]. It is also known that the "depth-first" algorithms do not guarantee the reaching of the goal [8].

There is common difficulty for the methods planning trajectory in the presence of known obstacles: it is very difficult to borrow full information about workspace of manipulator in advance and to represent this information in a form suitable for trajectory planning. Considering our algorithm one may see that there is no need for the control system to have full information about workspace in advance, manipulator will borrow necessary information by itself in limited quantities and in terms of generalized coordinates which is suitable for trajectory planning.

The attempts of creating algorithms for the robot control in presence of unknown obstacles were made. Most of them cover various two-dimensional cases [11].

In [11] the algorithm for the control of manipulators in the presence of unknown obstacles in three-dimensional space is given. Though this algorithm guarantees the reaching of a target position it has such a limitation that the manipulator should not have more than three degrees of freedom.

In [14] the n-dimensional case is considered. The algorithm is based on the solution of the system of nonlinear equations using Newton method and therefore it cannot guarantee the reaching of a target position.

2. Task Formulation and Algorithm

2.1. Preliminary Information

We will consider manipulators which consist of n rigid bodies (called links) connected in series by either revolute or prismatic joints [9]. The movement of an (i+1)-th link with respect to an i-th link in an i-th kinematical pair is described by the generalized coordinate $q_i(t)$. Vector $\mathbf{q}(t)=(q_1(t), q_2(t), \ldots, q_n(t))$ is called vector of generalized coordinates. If we know the functions comprising vector $\mathbf{q}(t)$ then we have complete information about how manipulator moves. In robotics generalized coordinate space is widely used. For a certain moment of time t^* manipulator is represented by a point in the generalized coordinate space $\mathbf{q}(t^*)=(q_1(t^*), q_2(t^*), \ldots, q_n(t^*))$. This point corresponds to a certain configuration of the manipulator in the Cartesian space and therefore the generalized coordinate space is also called configuration space.

In our case it will be necessary to move a manipulator from a start configuration to a target one. Let us denote start configuration as $\mathbf{q}^0=(q_1^0, q_2^0, \ldots, q_n^0)$ and target configuration as $\mathbf{q}^T=(q_1^T, q_2^T, \ldots, q_n^T)$. A trajectory $\mathbf{q}(t)$ is represented as a line in the generalized coordinate space and the movement of manipulator along this trajectory - as the movement of a point along this line from the point $\mathbf{q}^0$ to the point $\mathbf{q}^T$.

In our case the manipulator will have to move amidst unknown obstacles. The number of the obstacles, their disposition and shapes may be arbitrary. If the manipulator has at least one common point with any obstacle then such configuration of the manipulator will be considered as forbidden and the point $\mathbf{q}$ describing this configuration in the generalized coordinate space will be also considered as forbidden. If the manipulator has no common points with any obstacle then such configuration of the manipulator will be considered as permitted and the point $\mathbf{q}$ describing this configuration in the generalized coordinate space will be also considered as permitted. So the forbidden configurations will be represented in the generalized coordinate space as points but before the beginning of the movement the manipulator does not have information about them.

We must take into account that because of manipulator's constructive limitations vector-function $\mathbf{q}(t)$ must satisfy the set of inequalities

$$\mathbf{a}^1 \le \mathbf{q}(t) \le \mathbf{a}^2 \qquad (1)$$

for every time moment, where $\mathbf{a}^1$ is the vector of lower limitations on the values of generalized coordinates comprising $\mathbf{q}(t)$ and $\mathbf{a}^2$ is the vector of higher limitations.

The points satisfying the inequalities (1) comprise a hyperparallelepiped in the generalized coordinate space. We will consider all points in the generalized coordinate space which do not satisfy the inequalities (1) as forbidden too.

So in our problem a manipulator will be represented as a point which will have to move in the hyperparallelepiped from $\mathbf{q}^0$ to $\mathbf{q}^T$ and the trajectory of this point should not intersect with the forbidden points.

2.2. Task Formulation

Consider the problem of the manipulator movement amidst unknown obstacles in the following way: given a start configuration $\mathbf{q}^0$ and a target configuration $\mathbf{q}^T$, in the workspace obstacles may be present, but before the beginning of the movement there is no information about the presence of the obstacles, their number, disposition and shapes. It is necessary to make the manipulator move from $\mathbf{q}^0$ to $\mathbf{q}^T$.

Let us make the following considerations:

1) The disposition, shapes and dimensions of the obstacles do not change during the whole period of the manipulator movement;

2) It is known in advance, that the target configuration is reachable (that is, we know that in the generalized coordinates space it is possible to find such a line connecting $\mathbf{q}^0$ и $\mathbf{q}^T$ that this line will not intersect with any forbidden point);

3) The result trajectory must satisfy the inequalities (1) for every time moment. That is the whole movement must take place inside the workspace;

4) The manipulator has a sensor system which allows to define whether the manipulator intersects with an obstacle or not for an arbitrary current configuration $\mathbf{q}$ and for all configurations lying in a small r-neighborhood of $\mathbf{q}$. r-neighborhood is a hyperball in the generalized coordinates space with the center in $\mathbf{q}$ and with a radius r>0. We will not consider the structure of the sensor system.

5) We denote a set of all configurations from a r-neighborhood of $\mathbf{q}$ as $Y(\mathbf{q})$. The set of all forbidden points from $Y(\mathbf{q})$ will be denoted as $Q(\mathbf{q})$, the set of all permitted points from $Y(\mathbf{q})$ will be denoted as $Z(\mathbf{q})$. An r-neighborhood of $\mathbf{q}$ with the sets $Z(\mathbf{q})$ and $Q(\mathbf{q})$ may look in the following way (Figure 1):

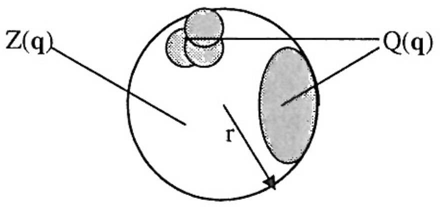

Figure 1.

Note that the sets $Z(\mathbf{q})$ and $Q(\mathbf{q})$ may be not continuous.

2.3. Algorithm

The algorithm of manipulator movement amidst unknown obstacles given below is the modification of the algorithm given in [7, 8]. Our algorithm is as follows:

1. Consider that a manipulator is in a start configuration $\mathbf{q}^0$. Manipulator investigates a small r-neighborhood of the $\mathbf{q}^0$ and forms the sets $Z(\mathbf{q}^0)$ and $Q(\mathbf{q}^0)$. Manipulator generates a trajectory $\mathbf{q}(t)$, satisfying the following conditions:

1.1.It connects $\mathbf{q}^0$ and $\mathbf{q}^T$,

1.2.No point from this trajectory does coincide with any point from $Q(\mathbf{q}^0)$

1.3.It satisfies the inequalities (1)

The manipulator begins to follow this trajectory.
2.While following this trajectory two results may happen:
2.1.The manipulator will not meet any obstacle and it will reach the $\mathbf{q}^T$. After reaching the $\mathbf{q}^T$ the algorithm terminates its work.
2.2.The manipulator will come to such a point (let us denote it as $\mathbf{q}^n$, n=1,2,...) that the next point of the manipulator trajectory will coincide with a forbidden point.

In this case the manipulator, being in $\mathbf{q}^n$, investigates the r-neighborhood of the $\mathbf{q}^n$ and forms the sets $Z(\mathbf{q}^n)$ and $Q(\mathbf{q}^n)$. After that the manipulator generates a trajectory satisfying the following conditions:
2.2.1.It connects $\mathbf{q}^n$ и $\mathbf{q}^T$
2.2.2.No point from this trajectory does coincide with any point from the sets $Q(\mathbf{q}^i)$, i=0,1,...,n
2.2.3.It satisfies the inequalities (1)
The manipulator begins to move along the new trajectory, n increments by 1 and the algorithm goes to the item 2.

2.4. Theorem

Theorem. If a manipulator moves according to the algorithm given above it will reach the target configuration in a finite number of steps.

Proof. Suppose that the manipulator being in $\mathbf{q}^n$ (n=0,1,...), generated a trajectory, leading to $\mathbf{q}^T$ and began to follow this trajectory. If the manipulator does not meet obstacles it will reach the target configuration in finite number of steps (because the length of the trajectory is finite). Therefore, the endlessness of the manipulator wandering may be caused only by the endless repeating of the situation described in the item 2.2 of the Algorithm and therefore the endless generating of new trajectories may be caused by the two reasons:
1) the manipulator will infinitely return to the same point of the trajectory changing,
2) the number of points where it is necessary to change trajectory will be infinite.

Let us prove that all points where the manipulator changes its trajectory will be different. Suppose that the manipulator changed a trajectory being in a point $\mathbf{q}^s$, and later it again changed a trajectory, being in a point $\mathbf{q}^p$, that is s<p. Let us show that $\mathbf{q}^s \neq \mathbf{q}^p$. Suppose, at first, that, on the contrary, $\mathbf{q}^s = \mathbf{q}^p$. Then $Q(\mathbf{q}^s)=Q(\mathbf{q}^p)$. When the manipulator was in $\mathbf{q}^s$, it generated a trajectory which did not intersect with the sets $Q(\mathbf{q}^i)$, i=0, 1, ... , s. When the manipulator reached the point $\mathbf{q}^p$, it discovered that it was necessary to change the trajectory that is this trajectory intersected with the set $Q(\mathbf{q}^p)$. But $Q(\mathbf{q}^p)=Q(\mathbf{q}^s)$ and $Q(\mathbf{q}^s)$ was taken into account when this trajectory was generated. It means that the manipulator can not come to a point of the trajectory changing $\mathbf{q}^p$ which will be equal to any other point of the trajectory changing and it means that all points where the manipulator changes its trajectory are different.

Now let us show that the number of such points is finite. Suppose that it is infinite. All points of a trajectory changing must satisfy the inequalities (1). It means, that the sequence of these points is bounded. According to the Boltsano-Weierstrass theorem it is possible to extract from this sequence a convergent subsequence $\mathbf{q}^i$, i=1,2,... According to Cauchy property of the convergent sequences it is possible for any ε to find such a number s that all points $\mathbf{q}^i$, i>s will lie in an ε-neighborhood of $\mathbf{q}^s$. Let us take ε<r. Consider an arbitrary point $\mathbf{q}^i$ of the trajectory changing lying in the ε-neighborhood of $\mathbf{q}^s$. As far as in $\mathbf{q}^i$ the manipulator had to change the trajectory, it means that that trajectory intersected with $Q(\mathbf{q}^s)$ (because $\mathbf{q}^i$ and its neighbour points belong to $Q(\mathbf{q}^s)$). From this fact it is possible to make the conclusion that the set $Q(\mathbf{q}^s)$ was not taken into account when that trajectory was generated. But such situation is impossible if we strictly follow the conditions of the algorithm. The situation when a trajectory changing point belongs to the ε-neighborhood of another trajectory changing point will necessarily appear if the number of the points where trajectory is changing is infinite. But we showed that such situation is impossible and it means that a number of the points where it is necessary to change trajectory will be finite. The theorem is proved.

So, the number of cases when the manipulator has to change its trajectory and generate a new one is finite. It is possible to see that to generate a new trajectory means to find a trajectory in presence of known obstacles. Therefore we may also see that the problem of a trajectory finding in the presence of unknown obstacles may be reduced to the solution of the finite number of the tasks of a trajectory finding in the presence of known obstacles. Such methods are represented for example in [1, 4, 5, 13].

2.5. Theorem consequence

If the manipulator is situated in $\mathbf{q}^n$, where it is necessary to change trajectory, it is not obligatory to generate a new trajectory immediately but it is also possible to make at first a finite number of steps according to another algorithms. Then this finite number of steps will be added to the finite number of steps of the movement according to the basic algorithm and in sum we get the finite number of steps [7]. This consequence allows us to use algorithms which will help to escape from dead ends and simplify the process of a manipulator movement.

3. Simplified Algorithm

An algorithm which is given below is a representative of the family of algorithms using the theorem consequence [7]. In this algorithm $\mathbf{q}^n$ denotes a current configuration of the manipulator. Before the beginning of the algorithm functioning we open a stack where we will store the configurations in which the manipulator has changed its trajectory. The number of configurations which are in the stack is denoted by J. Before the beginning of the algorithm work the stack is empty and J=0.

Step 1. Consider that the manipulator is in the start configuration $\mathbf{q}^0$. n=0, J=0.

Step 2. If the target configuration $\mathbf{q}^T$ is reached, then the algorithm terminates its work. Otherwise calculate a route $\mathbf{q}^n\mathbf{q}^T$ connecting $\mathbf{q}^n$ and $\mathbf{q}^T$. $\mathbf{q}^n\mathbf{q}^T$ is a finite number of points lying on a straight line segment connecting $\mathbf{q}^n$ and $\mathbf{q}^T$. Store the number of points comprising $\mathbf{q}^n\mathbf{q}^T$ in a variable number_of_points.
Step 3. Form the sets $Y(\mathbf{q}^n)$, $Q(\mathbf{q}^n)$, $Z(\mathbf{q}^n)$.
Step 4. If there is no point from $\mathbf{q}^n\mathbf{q}^T$ which coincides with any point from $\cup Q(\mathbf{q}^i)$, i=0,1,..,n, then the manipulator moves to the next point of $\mathbf{q}^n\mathbf{q}^T$, n increments by 1 and the algorithm goes to the Step 5. Otherwise the algorithm goes to the Step 7.
Step 5. If n=1, make the stack empty and consider J=0.
Step 6. If n≤number_of_points then the algorithm goes to the Step 3. Otherwise the algorithm goes to the Step 2.
Step 7. Add $\mathbf{q}^n$ to the stack, increment J by 1. Choose a $\mathbf{q}\in Z(\mathbf{q}^n)$ according to the "pushing from the prehistory algorithm"(PPA, see below). Manipulator moves to $\mathbf{q}$, n increments by 1. Denote $\mathbf{q}$ as $\mathbf{q}^n$. The algorithm goes to the Step 2.

Note that in Step 5 we make the stack empty because n became equal to 1, it means that the manipulator began to move along preliminary trajectory and we consider it as a sign that the manipulator has escaped the dead-end.

In this algorithm the preliminary trajectory is chosen as a straight line segment in the generalized coordinate space. It significantly simplifies the calculation of the preliminary trajectory, though it also leads to that fact, that the reaching of the target configuration will not be guaranteed. Therefore the algorithm given above should be completed by the following condition: if the Step 7 was fulfilled N_7 times in sequence, then as a preliminary trajectory should be chosen a higher order line, which would be able to go round the forbidden points. Though it should be mentioned that the experiments demonstrated the sufficient effectiveness of algorithm - the target configuration as a rule, was reached and there was no need for the introduction of the higher order line.

We mentioned the "pushing from the prehistory algorithm" (PPA). It is used in order to simplify an escape from a dead end, which may arise in a r-neighborhood of a current configuration.

PPA is based on the following heuristical approach. In case we discover in Step 4 that $\mathbf{q}^n\mathbf{q}^T$ intersects with $\cup Q(\mathbf{q}^i)$, i=0,1,..,n, it means that the manipulator should move not to the next point of $\mathbf{q}^n\mathbf{q}^T$ but to another point $\mathbf{q}$ of the manipulator's workspace. This point must be not forbidden and therefore must be $\mathbf{q}\in Z(\mathbf{q}^n)$. We keep in the stack the manipulator's trajectory changing points. It means that near these points obstacles were discovered. Therefore we must choose such a point from $Z(\mathbf{q}^n)$ which lies as far as possible from the points from the stack and as close as possible to the $\mathbf{q}^T$. Before the beginning of the PPA functioning let us store in a variable number_of_configurations the number of configurations which are in $Z(\mathbf{q}^n)$. J is again the number of configurations in the stack. The PPA is as follows:
Step 1. Consider j=1.
Step 2. If j>J, then go to the Step 4. Otherwise consider $\mathbf{q}^j$ from the stack.
Step 3. Call Algorithm1 (see below) and get from it the modified $Z(\mathbf{q}^n)$ and the number of configurations in it in the variable number_of_configurations. Increment j by 1. Go to Step 2.
Step 4. Choose from $Z(\mathbf{q}^n)$ such a configuration $\mathbf{q}$ whose distance to the $\mathbf{q}^T$ is minimal in comparison with other configurations from $Z(\mathbf{q}^n)$.
Algorithm1. This algorithm receives from the callee $Z(\mathbf{q}^n)$, number_of_configurations, $\mathbf{q}^j$ and $\mathbf{q}^n$ and leaves in $Z(\mathbf{q}^n)$ those configurations whose distances to $\mathbf{q}^j$ are bigger than the distance between $\mathbf{q}^j$ and $\mathbf{q}^n$. After terminating its work Algorithm1 returns to the callee the modified set $Z(\mathbf{q}^n)$ and number_of_configurations. If upon the terminating of its work Algorithm1 discoveres that it threw all configurations from $Z(\mathbf{q}^n)$ then it returns $Z(\mathbf{q}^n)$ and number_of_configurations in the state they were on the moment of the calling of Algorithm1.

3.1. The Simplified Algorithm Implementation Results

The software for the imitating of a manipulator movement in the presence of unknown obstacles was created. The program is based on the simplified algorithm given above.

As an example let us take the imitation of the movement of the manipulator with three revolute joints. The manipulator moves in the plane xOy of the basic coordinate system. In its start configuration the manipulator lies on the positive semiaxis of the x axis, in the target configuration the manipulator should lie on the negative semiaxis of the x axis. There is one obstacle - a parallelepiped. Its faces are parallel to the planes of the basic coordinate system. Figure 2 reproduces the process of the manipulator movement. The target configuration was reached in 41 steps. The experiments with another manipulators and obstacles were also carried out for the simplified algorithm. They demonstrated the sufficient effectiveness of this algorithm.

The software for the control of the manipulator "Adept" in the presence of unknown obstacles based on the simplified algorithm was also created [10]. The experiments demonstrated sufficient effectiveness of the software.

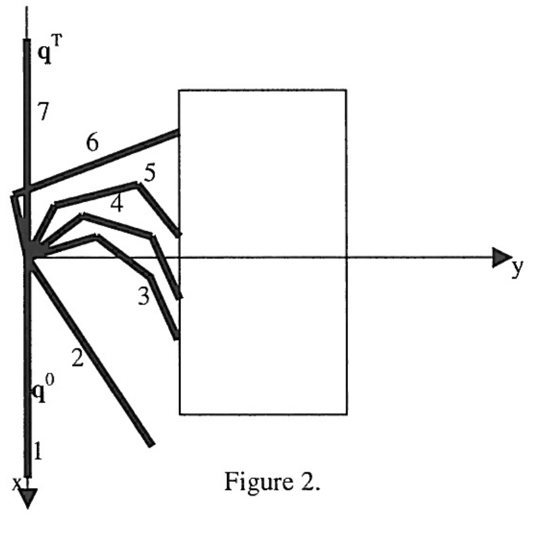

Figure 2.

The theory and algorithms given in this report were used in the russian scientific and research interuniversity program “Mechanics, engineering science and control processes” [3] for the creating of space robots.

4. Conclusion

The algorithm for a manipulator movement amidst unknown obstacles was presented. This algorithm is applicable for manipulators with n degrees of freedom working in the presence of unknown obstacles which may have arbitrary disposition, shapes and dimensions. Given the proof of the theorem stating that following this algorithm a manipulator will reach a target configuration in a finite number of steps. It is possible to see that in case of our algorithm there is no need to enter the full information about manipulator's environment in the manipulator control system: the manipulator gathers this information by itself and in quantities, sufficient for the target configuration reaching. Also was given the theorem consequence and the simplified algorithm using it. The results of the practical implementation of the simplified algorithm were given.

References

[1] C. Ahrikhencheikh, A. Seireg, Optimized-Motion Planning: Theory And Implementation, John Wiley & Sons, Inc, 1994.
[2] J. Barraquand, J.-C. Latombe, "Robot Motion Planning: A Distributed Representation Approach," Int. J. of Rob. Res., Vol.10, №6, pp.628-649, December 1991.
[3] Bernatsky I.P., Lopatin P.K. Issledovaniye Intellektualnykh Sistem Upravleniya Manipulatsionnymi Robotami V Neizvestnykh Ekstremalnykh Sredah. Otchet o nauchno-issledovatelskoy rabote po mezhvuzovskoy nauchno-tehnicheskoy programme “Mehanika, mashinovedeniye i processy upravleniya”. №01.20.00 13585. NII SUVPT, Krasnoyarsk. 177p., applications 81p. – manuscript.
[4] J. Canny, The Complexity Of Robot Motion Planning, The MIT Press. Cambridge, Massachusetts, 1988.
[5] G.E. Collins "Quantifier Elimination For Real Closed Fields By Cylindrical Algebraic Decomposition," Lecture Notes in Computer Science, Vol.33, pp.135-183, Springer-Verlag, New York, 1975.
[6] B.R.Donald, "On Motion Planning with Six Degrees of Freedom: Solving the Intersection Problems in Configuration Space," Proceedings of the IEEE International Conference on Robotics and Automation (1985)
[7] V.A. Ilyin, Algoritmy Planirovaniya Povedeniya Integralnykh Robotov V Usloviyakh Nepolnoy Informacii O Structure Vneshney Sredy. Tomsk, Izdatelstvo Tomskogo Universiteta, 1990.
[8] V.A. Ilyin, Intellektualnye Roboty: Teoriya I Algoritmy. Krasnoyarsk, SAA, 1995.
[9] C.S.G. Lee "Robot Arm Kinematics, Dynamics and Control," CompSAC 82: Proc. IEEE Comput. Soc. 6-th Int. Comput. Software And Appl. Conf., Chicago, Ill., Nov.8-12, 1982 - pp.601-610.
[10] P.K.Lopatin "Algoritm Dvizheniya Manipulatsionnogo Robota 'ADEPT' V Srede S Neizvestnymi Prepyatstviyami," Reshetnyovskiye Chteniya: Tezisy Dokladov IV Vserossiyskoy Nauchno-Prakticheskoy Konferencii Studentov, Aspirantov I Molodykh Specialistov (10-12 noyabrya, 2000, Krasnoyarsk) - Krasnoyarsk: SAA, 2000 -pp.206-207.
[11] V.J. Lumelsky, K. Sun "Three-Dimensional Motion Planning In An Unknown Environment For Robot Arm Manipulators With Revolute Or Sliding Joints," International. Journal of Robotics and Automation, Vol. 9, №4, pp.188-198, 1994.
[12] N.Nilson, Problem-Solving Methods in Artificial Intelligence. McGraw-Hill Book Company, New York, 1971.
[13] J. Schwartz, J. Hopcroft and M. Sharir (eds.), Planning, Geometry And Complexity Of Robot Motion Planning, Albex Publishing Co., New Jersey, 1987.
[14] F. Yegenoglu, A.M. Erkmen, H.E. Stephanou, "On-line Path Planning Under Uncertainty," Proc. 27th IEEE Conf. Decis. and Contr., Austin, Tex., Dec.7-9, 1988. Vol.2, pp.1075-1079, New York (N.Y.), 1988.