PLANNING ROBOT MOTION IN A 2-D REGION WITH UNKNOWN OBSTACLES

CARO LUCAS

Department of Electrical Engineering

Tehran University

Tehran, Iran

Received may 1988

Abstract The purpose of this paper is to present several algorithms for planning the motion of a robot in a two-dimensional region having obstacles whose shapes and locations are unknown. The convergence and efficiency of the algorithms are discussed and upper bounds for the lengths of paths generated by the different algorithms are derived and compared.

چکیده هدف این مقاله ارائه چند الگوریتم برای برنامه ریزی حرکت یك آدمك مصنوعی در یك ناحیه دوبعدی دارای موانعی با شکل و مکان نامعلوم است. همگرائی و کارآئی الگوریتمها مورد بحث قرار می‌گیرد و حدود بالا برای طول مسیرهای تولید شده توسط الگوریتم‌های مختلف محاسبه می‌شوند و مورد مقایسه قرار می‌گیرند.

INTRODUCTION

In recent years, the question of planning the motion of a robot system in the presence of obstacles which are not passable has been studied by many authors. It is usually converted to the geometric problem of finding a continuous path from the starting point S to the terminal point T that avoids the areas occupied by the obstacles [1-5]. In most of the published works on the motion planning of a robot, it has been assumed that full information about the environment and the positions of the obstacles is available. In this paper however, it is assumed that no information about the geometrical shapes and locations of the obstacles is initially available and the robot learns about the obstacles only as a result of meeting them along its path. The formulation of the problem is given in reference 6. The robot is modelled as a point located in a two dimensional plane and the obstacles are modelled as fixed and stationary regions. It is also assumed that there is a finite number of obstacles, any obstacle has finite area and perimeter, and any straight line can cross the boundary of an obstacle a finite number of times. The robot can recognize an obstacle when it encounters one and is able to move along the boundary of an obstacle or leave it so as to move freely in the area that is not occupied by the obstacles (the free space). The points S and T are known and the robot is capable of knowing its exact location and orienting itself in the two dimensional plane as well as moving along a straight line in the free space.

Two nonheuristic path planning algorithms have been given in reference 6. The upper bound for the length of paths generated by each algorithm has been considered as an indicator of the efficiency of that algorithm. It has also been shown that the least value for the upper bounds (called the "lower bound") is given by

$$LB = \overline{ST} + \sum_{i \in C_I} P_i \qquad (1)$$

where P_i is the perimeter of obstacle i, C_I is the set of all obstacles intersecting the ST-line and ST is the distance between S and T. An

improved algorithm having a smaller upper bound has been given in reference 7. The principle behind these algorithms is either to follow the $\overline{ST}$-line whenever moving in the free space or to leave the obstacles at the points of minimal distance from T. In this paper, three more algorithms based on a different principle are presented. It is argued that although these algorithms do not have smaller upper bounds than those previoiusly presented [6,7], they are more reasonable and leave the boundary earlier.

THE PATH PLANNING ALGORITHMS

Unless additional assumptions are made, no algorithm has so far been proposed whose upper bound is equal to the value of LB given by Lozano-Perez [1]. It has been claimed that the following algorithm is "the best that can be offered today" [6].

Algorithm 1

Initially, $j=1$; $L_0 = S$

I) From point L_{j-1} move toward T along a straight line until one of the following occurs:

a) Point T is reached. The procedure stops.

b) An obstacle is encountered and a hit point H_j is defined. Go to step 2.

2) Using the conventional direction (arbitrarily assumed left in the rest of this paper), follow the obstacle boundary. If point T is reached, stop. Use the registers R_1 to store the coordinates of the current point, Q_m of a minimal distance from the point T, R_2 to integrate the length of the boundary starting at H_j, and R_3, to integrate the length of the boundary starting at Q_m. (In the case of many choices for Q_m choose any). After having traversed the whole boundary and having returned at H_j, define a new leave point at $L_j=Q_m$. Go to step 3.

3) Using the contents of R_2 and R_3, determine the shorter way along the boundary to L_j. If the straight line segment L_jT is blocked by the obstacle at L_j, then the point T cannot be reached (test for reachability). Otherwise: set $j=j+1$ and go to step 1.

The upper bound on the length of paths generated by this algorithm is given by Lumelsky and Stepanov [8]:

$$UB_1 = \overline{ST} + 1.5 \sum_{i \in C} p_i \qquad (2)$$

where C_2 is the set of all obstacles intersecting or inside the circle C(T, ST) centered at T with radius $\overline{ST}$. Clearly $C_1 \subset C_2$. Actually, a better algorithm from the point of view of having the least upper bound, is given by Lucas [7].

Algorithm 2

Initially, $j=1$, $L_o=S$

1) From point L_{j-1} move toward T along the $\overline{ST}$-line until one of the following occurs:

a) Point T is reached. The procedure stops.

b) An obstacle is encountered and a hit point H_j is defined. Go to step 2.

2) Using the conventional direction, follow the obstacle boundary. If point T is reached, stop. Use the registers R_1 to store the coordinates of the intersection with the ST-line, Q_o currently having the least distance from the point T, R_2 to integrate the length of the boundary starting at Q_o. (In the case of many choices for Q choose any). After having traversed the whole boundary and having returned to Hj, define a new leave point as $L_j=Q_o$. (If the robot returns to Hj without ever meeting an intersection point with the ST-line then the point T cannot be reached). Go to step 3.

3) Using the contents of R_2 and R_3, determine the shorter way along the boundary to L_j. If the straight line segment L_jT is blocked by the obstacle at L_j then the point T cannot be reached. Otherwise, set $j=j+1$ and go to step 1.

The upper bound on the length of paths generated by Algorithm 2 is given by

$$UB \quad \overline{SA} + \quad 5 \sum_{i \in C1} p_i \qquad (3$$

which is smaller than UB_1. When an obstacle is encountered, the robot must traverse the

whole boundary under both algorithms, during which the "leave point" is decided. Then it must move along the boundary again so as to reach that leave point. The total length of path along the boundary generated by any algorithm will depend on the shape of the obstacle and one cannot determine, a priori,which algorithm will generate a shorter path. If no other obstacle is encountered after leaving the current obstacle, the remaining length of the path generated by Algorithm 1 will be shorter. But even in this case, one cannot determine, a priori, the total length of the path generated by which algorithm will be shorter. Therefore, Algorithm 2 is to be preferred because $UB_2 \leqslant UB_1$.

However, the upper bound alone is too conservative an index for determining the efficiency of an algorithm. Not all decision makers would prefer an algorithm having a smaller upper bound if in most cases of interest it generated longer paths. Furthermore, the bounds are expressed in terms of Pi, which depend on the locations and shapes of the obstacles. Generally, if an algorithm allows deviations from the ST-line there will be some possibility of encountering obstacles not belonging to C_1 (as well as obstacles belonging to C_1); and so, the upper bound, as expressed by P_i, will increase because more terms have to be included [7]. But one cannot say that there is always some likelihood of encountering additional obstacles because there is also some possibility of not encountering even those obstacles belonging to C_1.

The following algorithm can thus be proposed:

Algorithm 3

Initially, $j=1$, $L_0=S_i$

1) From point L_{j-1} move toward T along a straight line until one of the following occurs:

   a) Point T is reached. The procedure stops.

   b) An obstcle is encountered and a hit point H_j is defined. Go to step 2.

2) Using the conventional direction, follow the obstacle boundary. If point T is reached, stop. Use the registers R_1 to store the coordinates of the first point, Q_1 along the boundary such that Q_1T does not intererst the obstacle at any point and $Q_1T < H_jT$, R_2 to register the last point, Q_2 along the boundary having the same properties, R_3 to integrate the length of the path along the boundary from H_j to Q_1, and R_4 to integrate the length of the path along the boundary from Q_2 to H_j*. If the robot returns to H_j without having found any point with the properties given for Q_1 or Q_2, then point T cannot be reached (test for reachability). Otherwise go to step 3.

3) Using the contents of R_3 and R_4 determine the point Q_i (i=1 or 2) having the shorter distance along the boundary path from H_j. Define $L_j=Q_i$. Set $j=j+1$ and go to step 1.

The upper bound for this algorithm is equal to that of Algorithm 1.

$$UB_3=\overline{ST}=1.5\sum_{i\in I} P_i \qquad (4)$$

However, it can be easily seen that the path along the boundary under Algorithm 3 is shorter than under any of the other two algorithms; and if no other obstacle is met, then the total path generated by Algorithm 3 will be the shorter (Figure 1). We can, in fact, further improve Algorithm 3 by modifying R_3 and R_4 in step 2 to include the distance along the boundary from Hj to Q_i plus the distance Q_iT (Algorithm 4).

It is to be noted that the upper bounds corresponding to Algorithm 1, Algorithm 2, and Algorithm 3 are all greater than LB. This is due to the fact that the robot completes a close curve along the boundary of each obstacle it encounters before it determines the leave point of that obstacle, and then it has to move along the boundary once again (but chosing the shorter path, along the

* Further memorization (e.g. angles for the Q_iT-line) is necessary for this algorithm.

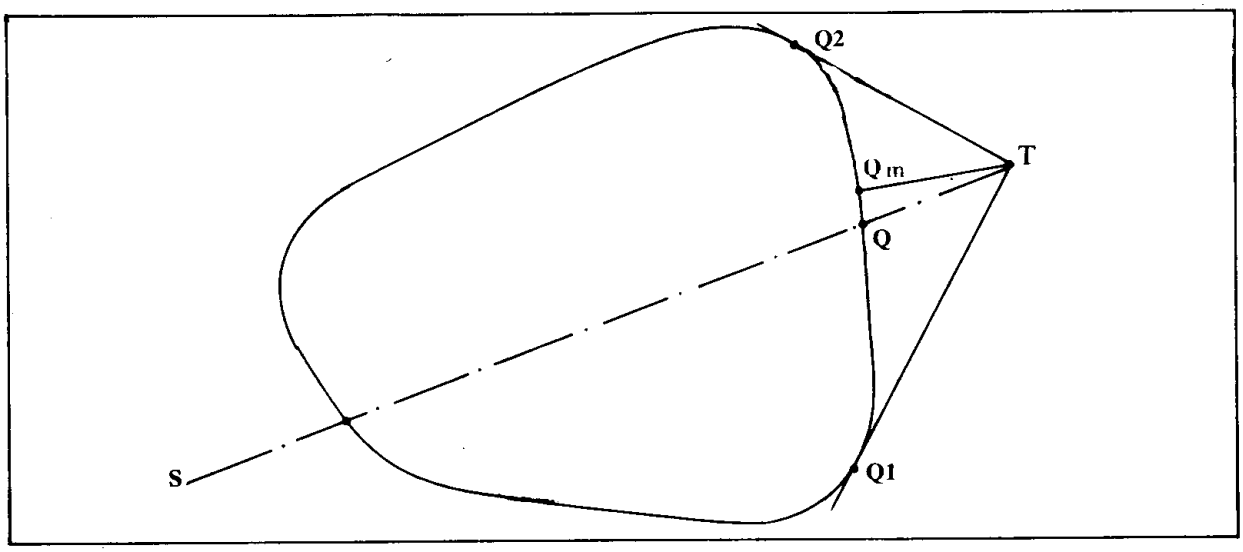

Figure 1: Comparison of the paths generated by Algorithm 1, Algorithm 2, and Algorithm 3.

boundary, from the hit point to the leave point) so as to reach the desired leave point. Thus, the total path along the boundary is always greater than P_i and in the worst case it is equal to $1.5P_i$. The following algorithm proposed in reference 6 avoids this drawback.

Algorithm 5

Initially, $j=1$, $L_0=S$

1) From point L_{j-1} move along the ST-line until one of the following occurs.
   a) Point T is reached, the procedure stops.
   b) An obstacle is encountered and a hit point H_j is defined. Go to step 2.
2) Using the conventional direction, follow the obstacle boundary. If point T is reached, stop. If the ST-line is met at a distance d from T such that $d< H_j T$, then define the point L_j, set $j=j+1$ and go to step 1. If the robot returns to H_j without ever meeting the ST- line, or before identifying another hit point H_{j+1}, then point T cannot be reached.

However, unless additional assumptions (such as convexity of the obstacles) are made, there is always the possibility of coming across the same obstacle after having left it at some other point, as well as having to pass some segment of the obstacle boundary several times. As for Algorithm 5, the upper bound can be expressed as

$$UB_4 = \overline{ST} + \sum_{i \in I} 1/2\, n_i P_i \qquad (5)$$

where n_i is the number of intersections between the ST-line and the i-th obstacle boundary. If, e.g., under convexity assumption, we have $n_i=2$; $i=1, 2,\ldots$, then UB_4 will be equal to LB. Finally, if we assume that all obstacles are convex, then the following algorithm may also be proposed.

Algorithm 6

Initially, $j=1$, $L_0=S$

1) From point L_{j-1} move toward T along a straight line until one off the following occurs.
   a) Point T is reached, the procedure steps.
   b) An obstacle is encountered and a hit point H_j is defined. Go to step 2.
2) Using the conventional direction, follow the obstacle boundary. If point T is reached, stop. If a point Q_1 is reached such that $Q_1T < H_jT$ and the straight line Q_1T is not blocked by the obstcle at Q_1, then set $j=j+1$, $L_j=Q_1$ and go to step 1.

Although the paths generated by this algorithm are not bounded by (5), because there is now the possibility of meeting additional obstacles not intersecting the ST-

line Algorithm 6 is not inferior in Algorithm 5. The possibility of meeting an increased number of obstacles does not necessarily imply an increased likelihood of coming across more obstacles, and there is also the possibility of avoiding some of the obstacles intersecting the ST-line without meeting other obstacles instead. On the other hand, if after leaving an obstacle neither algorithm generates a path that meets another obstacle, then Algorithm 6 will generate the shorter path.

CONCLUSION

After discussing the two algorithms proposed in [6], as well as the "optimal" algorithm given in [7] (optimal with respect to the "upper bound" criterion as suggested in [6]), new algorithms, which, have a greater upper bound than the "optimal" algorithm and are more efficient in certain other respects have been proposed here. Algorithms 3 and 4, while having the same upper bound as Algorithm 1, are more efficient than both Algorithms 1 and 2 in those other respects. Algorithm 6 combines the features of these Algorithms and Algorithm 5 so as to avoid investigating the whole boundary of each obstacle" (as suggested in [6]). However, additional assumptions were necessary for guaranteeing convergence and efficiency [7] . It has also been argued that the upper bounds, besides being a rather conservative criterion, are expressed in terms of the perimeters P_i of the obstacles, which are initially unknown and therefore, other criteria for efficiency should also be considered.

AKNOWLEDGEMENT

The author wishes to thank the reviewers of the journal for their careful analysis of the paper and their many useful suggestions. which have contributed to the improvement of the presentation and elimination of possible sources of error.

REFERENCES

1. T. Lozano-Perez. "Special Planning: A Configuration Space Approach" IEEE Trans. Comp. Vol. C-10, Feb. 1983.
2. K. C. Shin and N. D. McKay. "Robot Path Planning Using Dynamic Programming". IEEE Trans. Automat. Contr., Vol. AC-31, June 1986.
3. M. W. Spong, J. S. Thorp, and J. M. Kleinwaks. "The Control of Robot Manipulators with Bounded Imput". IEEE Trans. Automat Contr. Vol. AC-31, June 1986.
4. H. P. Moravec. "The Stanford Cart and the CMU Rover". Proc. IEEE No. 7, 1983.
5. R. A. Brooks. "Solving the Find-Path Problem by Representing Free Space as Generalized Cones". Artificial Intelligence Lab., MIT, Cambridge, AI Meno. 674, May 1982.
6. V. J. Lumelsky, adn A. A. Stepanov. "Dynamic Path Planning for a Mobile Automation with Limited Informatiion on the Environment". IEEE Trans. Automat. Contr., Vol. AE-31, November 1986.
7. C. Lucas. "Comments on Dynamic Path Planning for a Mobile Automation with Limited Information on the Environment". IEEE Trans. Automat. Contr. In Press.
8. V. J. Lumelsky and A. A. Stepanov. "Authors' Reply". IEEE Trans. Automat. Contr. In Press.