장애물이 있는 경우의 효율적인 로보트 동작계획

A Simple and Efficient Planning of Robot Motions with Obstacle Avoidance

정봉주, 이영훈
B.J. Jeong, Y.H. Lee

삼성전자 반도체부문 CIM팀
CIM Team, Semiconductor Business, Samsung Electronics Co.

ABSTRACT

This paper deals with the efficient planning of robot motions in the Cartesian space while avoiding the collision with obstacles. The motion planning problem is to find a path from the specified starting robot configuration that avoids collision with a known set of stationary obstacles. A simple and efficient algorithm was developed using "Backward" approach to solve this problem. The computational result was satisfactory enough to real problems.

1. Introduction

The motion planning problem, in its simplest form, is to find a path from a specified starting robot configuration that avoids collision with a known set of stationary obstacles. Note that this problem is significantly different from, and quite a bit harder than the collision decision problem: detecting whether a known set configuration or robot path would cause a collision. Motion path planning is also different from on-line obstacle avoidance: modifying a known robot path so as to avoid unforeseen obstacles. Most recent efforts have included obstacle avoidance strategies aimed at real-time robot control[1, 2] and near-optimal approaches[3] using potential function concepts to avoid obstacles. Mayne[4, 5] used the recursive quadratic programming (RQP) in motion planning and considered robot motions in allowable 2-D workspace.

Figure 1 shows the example of the workspace and robot configuration with 2 joints. We want the robot arm to move the current configuration H to the target point A, B, C, and D sequentially while avoiding the collision with obstacles OA, OB, OC, and OD. In this paper, the simple and efficient algorithm for the path planning will be presented using the geometric approach. We consider the polygonal workspace and obstacles. The obstacles may be non convex like OA.

2. Workspace and Robot Arm Representation

The polygonal workspace and obstacles are considered. The polygon can be represented as a set of vertices (usually counterclockwise). The sequence of vertices is needed to identify the inside area of polygon or outside. When we travel on edges of the polygon in the counterclockwise, the left side area is the inside of the polygon.

Workspace Representation

Since the workspace border and obstacles are polygons, they have the same representation scheme. The obstacle P is represented as follows.

- number of vertices: n
- sequence of vertices: (V1, V2, ..., Vn)
- coordinates of vertex: (X_i, Y_i) for vertex i
- coefficients for the edge equation: $(a_i, b_i, c_i)(a_i, b_i, c_i)$ for edge i
  where$a_i X + b_i Y + c_i = 0$ $a_i X + b_i Y + c_i = 0 (X_i, Y_i)$

Robot Arm Representation

The robot arm configuration C is represented as follows.

- number of joints: nj
- coordinate of each joint, end-effector and the base
  for the *i* th joint, J(i) = (JX(i), JY(i)) where i =1 to nj+1,

for the base, B = (bx, by).

- length of link: L(i) where i = 0, 1, ..., nj.
- length of robot arm: $r = \max_{i \in J} \{|J(i) - B|\} \max_{i \in J}$.
- angle between adjacent links: $COS(Ai) = (J(i)-J(i-1))(J(i+1)-J(i))/L(i-1)L(i)$

Figure 2 shows an obstacle and robot arm representation.

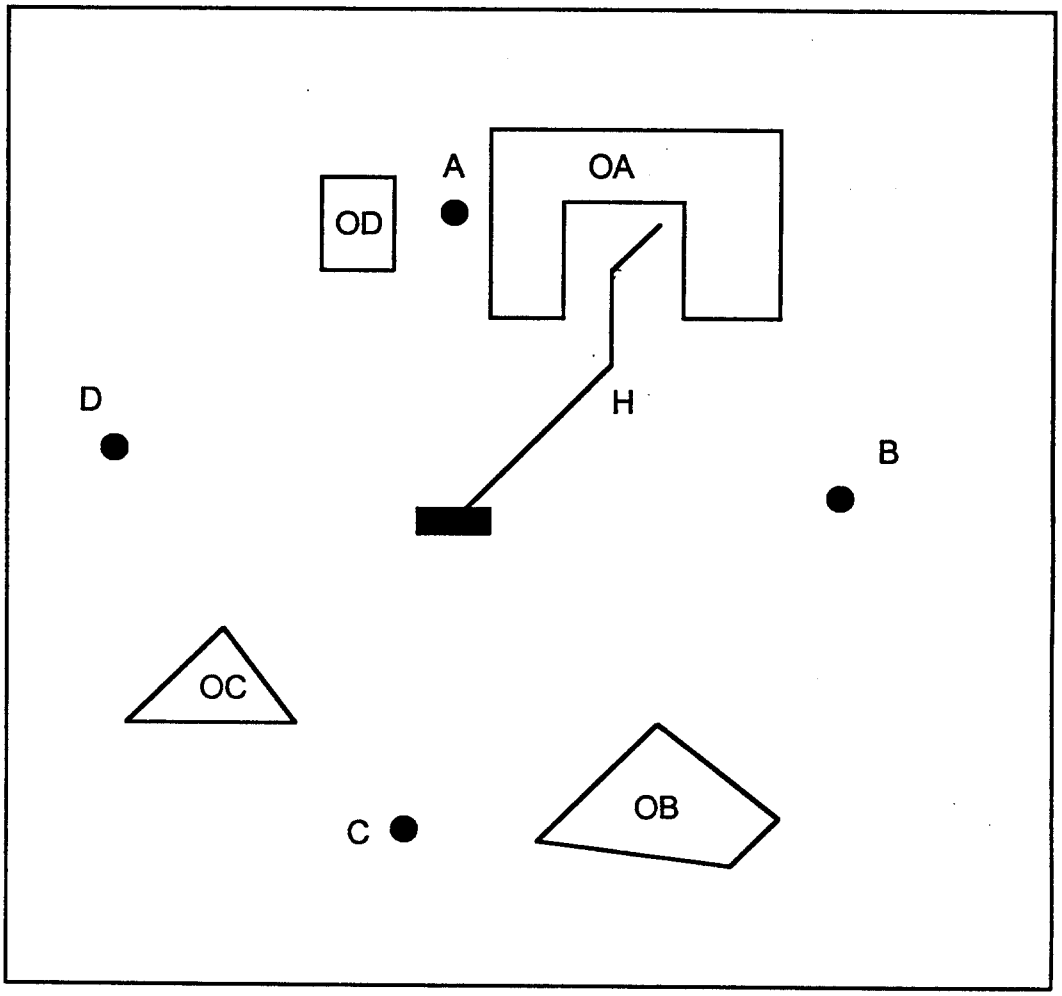
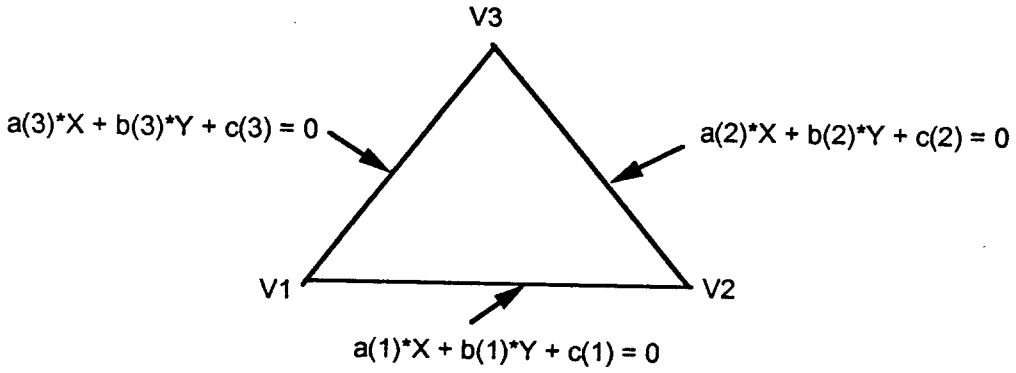
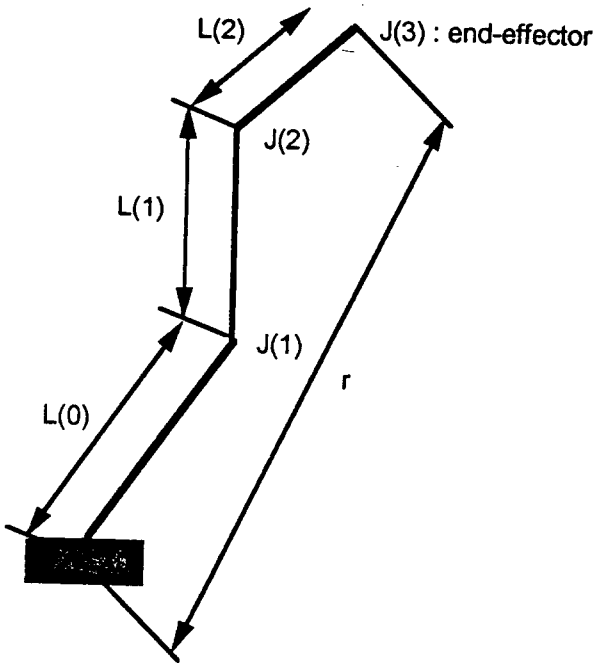


Figure 1. Workspace and Robot Arm Configuration



(a) Obstacle



(b) Robot Arm

Figure 2. Representation of Obstacle and Robot Arm

3. Algorithm

The most difficult problems in finding the motion path arise from the fact that there exist infinite solutions because the feasible solution space is the set of real number points. The basic concept of this algorithm is to reduce the solution space very effectively and consider only the efficient solution space. To do this, we use the "backward(or goal-directed)" approach to find the solution.

The algorithm consists of two stages. In the first stage, the algorithm finds the efficient robot arm configuration corresponding to each target point. These configurations are target configurations of the robot arm. In the next stage, the algorithm finds the feasible motion path from the initial configuration to target configurations corresponding to each target point.

3.1 Find the target configuration: algorithm TARGET_CONFIG

For a given target point for the end-effector, we want to know the feasible and efficient robot arm configuration. The basic idea of this algorithm is that starting with the target point, finds all joint positions by the reverse joint sequence, i.e., J(nj), J(nj-1), ... , J(2), J(1) which solves the following equations.

minimize $| J(2) - B |$
subject to $| J(i+1) - J(i) | = L(i)$ for i =1 to nj
all link lie in the free-space

Since J(1) can be computed after J(2) is determined, we want to find J(2) position which gives the shortest distance to the base.

Algorithm TARGET_CONFIG: find all feasible joint positions

Initialization: Let i← ← nj, J(nj+1) = target point where nj: number of joints
Step 1: From J(i+1), draw the visible path to the base.
If there is no path, find visible paths to the obstacles. Go to step 2.
Step 2: Find J(i) such that
$J(i) = J(i+1) + d^*t,\ t > 0$
$| J(i) - J(i+1) | = L(i)$
where d: unit visible direction vector
L(i): length of link i.
If i = 2, then go to step 3.
Otherwise, i ← i-1 and go to step 1.
Step 3: Find J(1) such that
$| J(2) - J(1) | = L(1)$
$| J(1) - B | = L(0)$
where J(1), J(2): joint 1(unknown) and joint 2 position
B: base position
L(0), L(1): length of link 0 and 1.
If found, stop. Otherwise, go to step 4.
Step 4: Let $A1 = \{(x,y): | J(2) - J(1) | < L(1)\}$, $A2 = \{(x,y): | J(1) - B | < L(0)\}$.
If $A1 \cap A2 = \varnothing$, then there is no feasible position, stop.
If $A1 \cap A2 = A2$, then i ← nj and choose the other feasible direction d*. Go to step 3.

3.2. Find the Motion Path: algorithm FIND_PATH

Definition

Cj: robot arm configuration at the end-effector point j.
Cj*: robot arm configuration which has the same joint angles as Cj.
C(r): robot arm configuration with arm length r.
r: current robot arm length = max $| J(i) - B |$ for i = 1 to nj+1
l: minimum length of robot arm
r*: minimum distance from base to obstacles.

Suppose we want the robot arm to move from A to B as shown in Figure 3. Algorithm FIND_PATH finds a feasible robot arm path from A to B point.

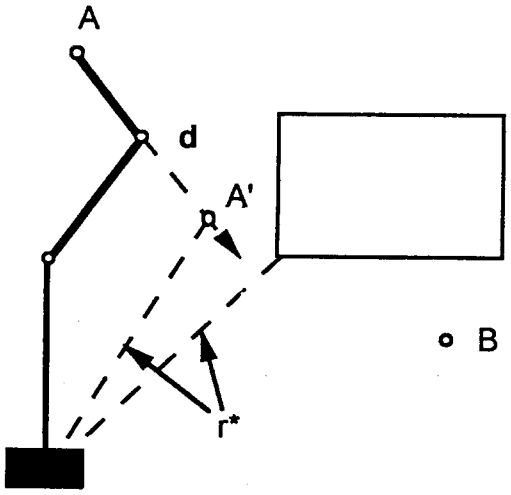


Figure 3. Make configuration C(r*) on the direction **d**

Algorithm FIND_PATH

Step 1: Generate visible path from the base to obstacles between A and B point.
Compute r*.

Step 2: Case 1: $r < r^*$
Rotate to the B and make $C_B C_B$.

Case 2: $l < r < r^*$
Find the position P with length r* on the direction d form the current end-effector to the adjacent joint position.
Make $C_P C_P(r^*)$ by using TARGET_CONFIG.
Rotate to B and make C_B.

Case 3: $r^* < l$
Infeasible.

4. An Illustrative Example

The workspace and robot configuration in Figure 1 is used as an example. Figure 4 shows all target configurations and the accumulated motion path from the initial point to target points.

5. Conclusion

In motion planning problem, there are too many free-space paths in Cartesian Space. In order to consider only the efficient motion path, we have to reduce the solution space. The simple and efficient algorithm was developed using "Backward" approach to do this. The algorithm finds target configurations corresponding to target points and then motion paths between the starting point and target points are generated. Experimental results show that it takes less than a second to find the motion path from one point to the next target point on the IBM 486 PC. The drawback of this algorithm is that it requires a conversion procedure which converts the Cartesian Space solution to Joint Space solution to implement on the robot. This algorithm can be extended to 3-dimensional problems by changing 2-D equations to 3-D's.

References

[1] W.E. Red and K.H. Kim, "Dynamic Direct Subspaces for Robot Path Planning," Robotica, Vol. 5, pp. 29-36, 1987.
[2] Thomas Lozano-Perez, 'A simple motion-planning Algorithm for General Robot Manipulators," IEEE Journal of Robotics and Automation, Vol. RA-3, June 1987.
[3] O. Khatib, "Real -time Obstacle Avoidance for Manipulators and Mobil Robots," The International Journal of Robotics Research, Vol. 5, No. 1, 1987.
[4] R.W. Mayne, 'position and Motion Planning in a Confined Region via Nonlinear Programming," The 1987 ASME Design Technology Conference, Boston, Mass. Sep. 27-30, 1987. Advances in Design Automation, Vol. 1, 1987.
[5] C.Y. Liu and P.W. Mayne, "Motion Planning and Geometric Design in a Two-dimensional Space," 1989 World Conference on Robotics Research: The Next Five Years and Beyond, Conference Proceedings, May 7-11, 1989. Robotics International of SME, 1989.

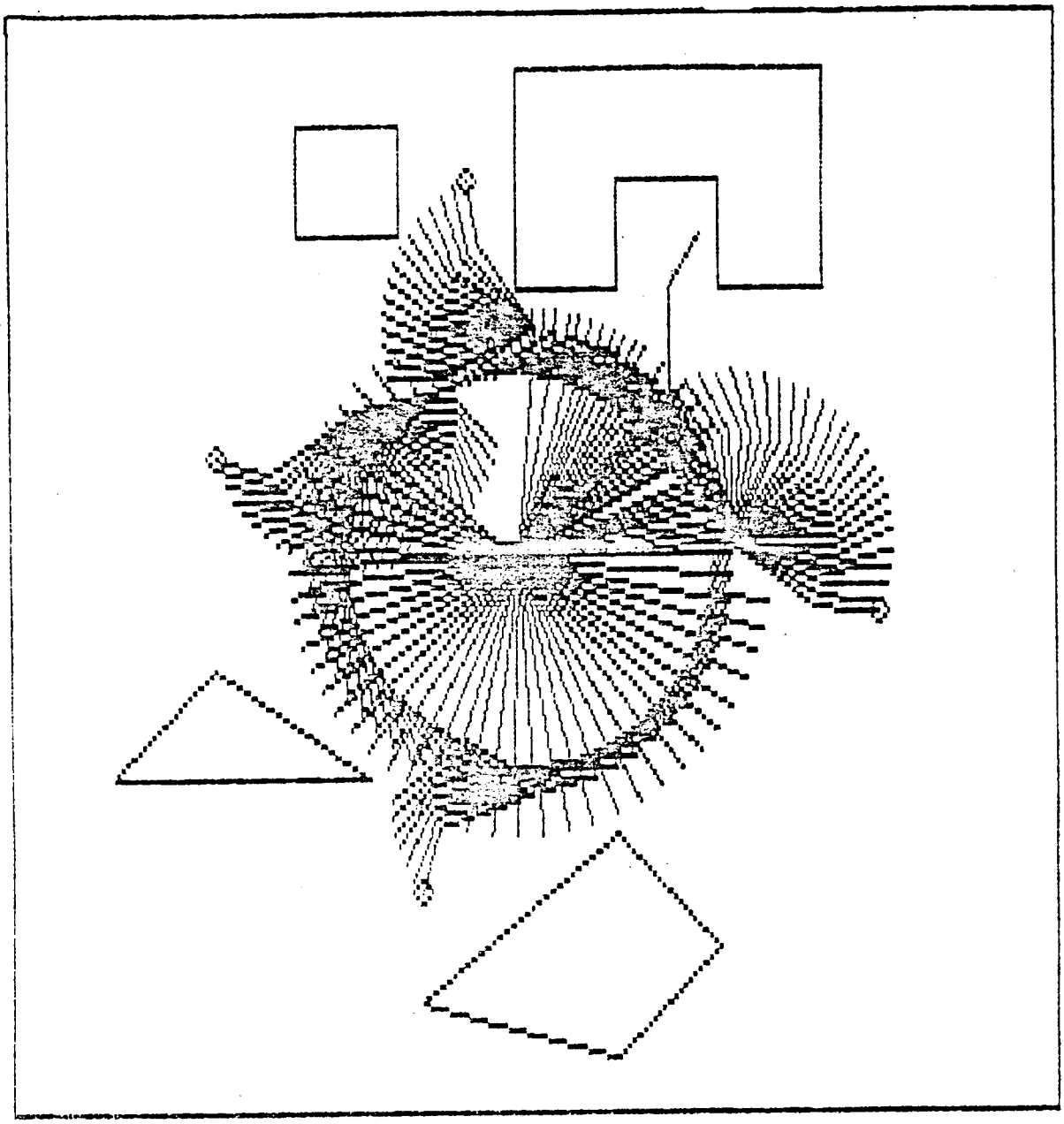


Figure 4. The Motion Paths of Robot Arm