Journal of University of Science and Technology Beijing
1998, Vol.5, No.2: 104

Automation

Autonomous Path Planning For Robot Manipulators in Manufacturing Environment

Wei Wang Yang Yang Kui Yuan
Information Engineering School, UST Beijing, Beijing 100083, China
(Received by 1997-11-27)

Abstract: A robot intelligent path planning system(RIPPS), which can be used as a robot off-line programming tool was introduced in this paper. C-space modeling and path search are key parts of the system. A fast C-space modeling is presented based on critical collision joint angle(CCJA), C-obstacle boundary can be computed by the feature points of obstacles. A modified A^* algorithm is used for path search, although the path is slightly sub-optimal, a tremendous speed increase is achieved. RIPPS runs on a Silicon Graphic 4D/70 workstation, simulation results show that the system is efficient and practical for robot collision-free autonomous path planning in a structured and known environment.
Key words: robot, path planning, A^* algorithm

Most industrial robot manipulators work by teaching operation and have little autonomous capability. The traditional teaching operation can not meet the demands of the advanced manufacturing technology and restricts the robots application in manufacturing. To make robots complete tasks with high intelligent capabilities in obstacle environment, a relevant effort is to develop autonomous capabilities of robots, such as collision-free motion planning. Many researchers have done much contributions in robot collision-free path planning, the methods developed include graph search[1~3], potential field method[4], control algorithm[5] and topologic method[6].

RIPPS was developed as an off-line programming tool, such as autonomous path planning and graphic monitoring of robot tasks for robot manipulators in manufacturing environment. A fast robot C-space modeling method based on critical collision joint angle is presented. Although manipulator arms are substituted by cylinders or rectangular solids as usual, the algorithm does not require that obstacles be expanded and the manipulator arms be shrunk to line segments, only the critical collision joint angles of feature points of obstacles are needed to computed. So the algorithm needs less computation and can be integrated with robot dynamic graphic simulation. To find a collision-free path in C-space, a modified A^* algorithm with dynamic changeable step and a goal-visible-test is employed, although the path is slightly sub-optimal, a tremendous speed increase is achieved.

RIPPS includes several modules: robot and environment modeling, graphic simulator, C-space modeling, path search. This paper will deal with C-space modeling in section 2 and section 3, path search in section 4, a simulation example is given in section 5.

1 2D C-space Modeling

C-space[1] path planning approach can be grossly described as being comprised of two phases of during which a data structure is first built that represents the geometric constrains imposed by the obstacles, and then search for a solution. Typically, the process of building the data structure consists of mapping the obstacles from the robots workspace into the robots C-space. The C-space is the joint space for robot manipulators. The result of this operation is an explicit representation of all robot configurations that do not result in a collision with some obstacle, that is the robots free space.

1.1 Critical collision joint angle(CCJA)

A planar linkage is considered here. To compute the boundary of the C-obstacle for link K, assume that the previous joints are fixed at certain joint angles. When link K collides with a point, there must have two critical angles: the first angle at which link K contacts the point when link K rotates in counterclockwise direction, we call the angle lower critical collision joint angle θ_{LC}. Another angle is similar to θ_{LC}, it is formed when link K rotates in clockwise direction and called upper critical joint angle θ_{UC}. They are illustrated in Fig. 1.

When link K touches a polygonal obstacle, there

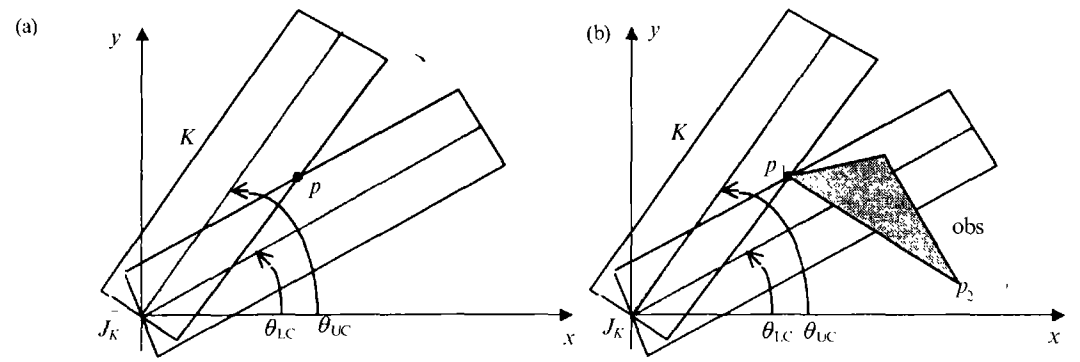

Fig.1 Upper critical collision joint angle, lower critical collision joint angle (a) and the valid critical joint angle (b) for point p

will be a collision point p, the two collision points p_1 and p_2 as shown in Fig. 1(b). We can compute θ_{LC} and θ_{UC} for a collision point, but only one CCJA is valid for a collision point, because the practical collision can not happen for another CCJA. In Fig. 1(b), for example, the θ_{UC} of point p_1 is valid and considered as the upper bound of the C-obstacle of the obstacle, and the θ_{LC} of point p_1 is invalid. For collision point p_2, θ_{LC} is valid and considered as the lower bound of the C-obstacle.

We define the possible collision points as feature points of obstacles. The feature points of obstacles include: (1) Link K rotates about its joint, the end link K has a circular path, the intersection points of the path with obstacle edges are feature points. (2) The vertices of the obstacle in Body-space of link K are feature points.

1.2 The boundary of 2D C-obstacle

Any C-obstacle has a closed curve boundary, i. e. C-boundary, and any C-boundary is formed by simple closed curves, the joint angles are the configuration parameters. If link L_j collide with an obstacle, the feature points FP_i of the obstacle are computed. For each FP_i, computing $\theta_{LC}(FP_i)$ and $\theta_{UC}(FP_i)$, then minimum $\theta_{LC}(FP_i)$ is lower bound point LP_j of C-obstacle corresponding the current pose of joint J, the maximum $\theta_{UC}(FP_i)$ as upper bound point UP_j of The C-obstacle. Computing every LP and UP for each discreted pose of joint J, then the lower boundary LCO(obs), and upper boundary UCO(obs) of the C-obstacle can be obtained by LP and UP respectively, i.e

$$LP_j = \min(\theta_{LC}(FP_i)) \quad (1)$$

$$UP_j = \max(\theta_{UC}(FP_i)) \quad (2)$$

$$LCO(obs) = \bigcup_{j=1}^{n} LP_j \quad (3)$$

$$UCO(obs) = \bigcup_{j=1}^{n} UP_j \quad (4)$$

Where, n is discreted point amount of joint J, the C-boundary BCO(obs) is

$$BCO(obs) = LCO(obs) \bigcup UCO(obs)$$

2 3D C-space Modeling

2.1 The tori approximation of obstacles

The environment objects are assumed to have vertical sides and horizontal or inclined top/bottom. Actually, many objects can be divided and considered as a composition of several polyhedral objects with vertical sides. An obstacle can be represented by a prism which is a convex hull of the real obstacle, this description can cause little loss of free space.

For robot manipulators, the links can be divided into two groups: the major linkage which positions the end-effector, and the minor linkage which orients the end-effector. For most commercial robot manipulators, the major linkage is a planar 2-DOF linkage, which is in a vertical plane or two parallel vertical planes V-plane named in this paper. According to the features, an obstacles can be approximated with a torus, which has four parameters: $\Delta\theta_1$, min$r$, max$r$, and ΔZ, shown in Fig. 2. This allows us to consider the arm as planar polygon and to operate in H-plane and V-plane, so path planning in 3D space can be accomplished by partitioning the first joint angle. The degrees of freedom are reduced to only three and the last three joints are considered as free and includes all their possible positions in the polyhedra of the robot hand. This method is a general and is applicable to most of the commercial robots, by simply changing the corresponding models and their kinematics trans-

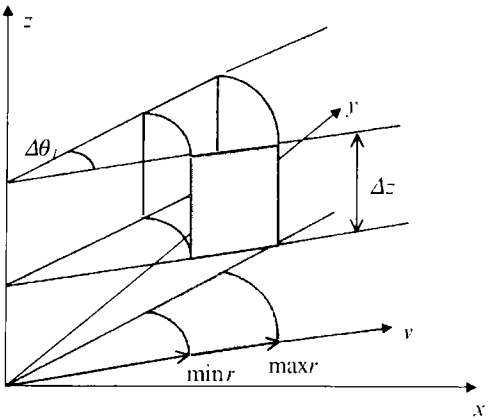

Fig.2 H-plane(xoy), V-plane(voz) and the torus approximating an obstacle

for-mations.

2.2 The boundary of 3D C-obstacle

The rotation of the first joint is divided into angular intervals in H-plane, V-plane may cut through the torus and will generate a rectangular cross-section(RCS) which extends for the angular interval determined by the torus in H-plane. For RCS, all or only a part of it is in the workspace, we must compute the part of RCS in workspace, which is named true cross-section(TCS), the boundary of TCS consists of line segments and arcs. In V-plane, we can calculate the possible collision points of TCS for joint 2 and joint 3. For every feature point, compute θ_{LC} and θ_{UC}, and get $\min(\theta_{LC})$ and $\max(\theta_{UC})$ as the lower bound point LP_i and upper bound point UP_i of the C-boundary respectively, calculate all LP_i and UP_i, the C-boundary consists of LP_i and LP_i. The procedure consists of the following steps

(1) Transform an obstacle to a torus;

(2) Get θ_1 interval $\Delta\theta_1$ in H − plane;

(3) Compute RCS in V − plane and obtain TCS from RCS;

(4) Compute all feature points by TCS;

(5) Compute θ_{LC} and θ_{UC} for every feature point;

(6) Get LP_i and UP_i of C-boundary by $\min(\theta_{LC})$ and $\max(\theta_{UC})$ respectively;

(7) Go to step 5, if over the joint limits, the procedure stops.

3 Path Search Algorithm

The A^* algorithm[7] is frequently employed for search-

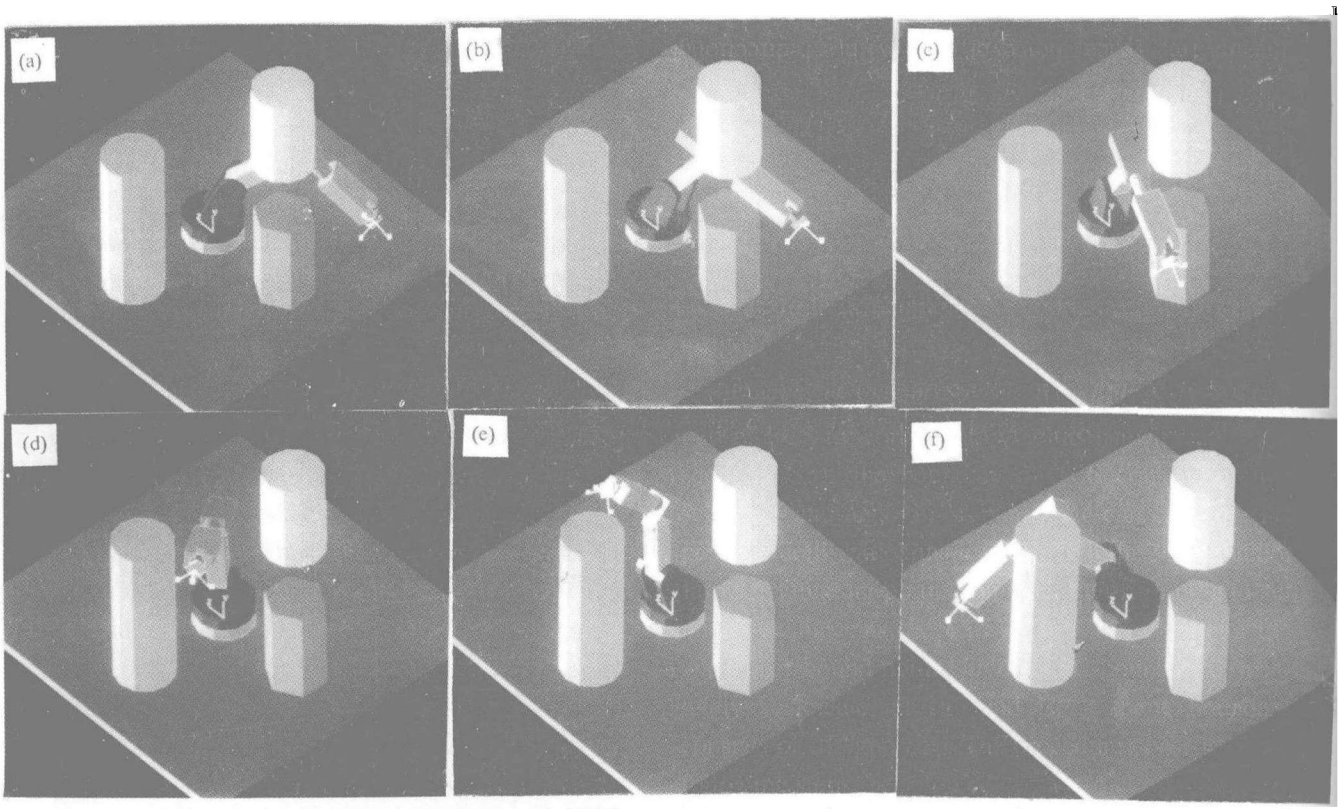

Fig.3 Collision-free motion of a MOTOMAN-K30S robot

ing the free space, however, even with a bidirectional search, the *A** turns out to be rather slow. The alternative presented here is to use dynamic step-changeable *A** algorithm with a goal-visible-test. With this modification, although the path is slightly sub-optimal, a tremendous speed increase is achieved. The algorithm is described broadly as follows:

(1) OPEN:=(START), f(START):=h(STAR);
(2) Loop: If OPEN=(NULL) THEN EXIT(FAIL);
(3) n:= FIRST(OPEN);
(4) REMOVE(n, OPEN), ADD(n, CLOSED);
(5) IF VISIBLE(GOAL), THEN EXIT(SUCCESS);
(6) k:= MaxStep;
(7) DynamicStep= kBasicStep;
(8) If NO(SCS(n)), THEN k:=k − 1, Go to (7);
(9) EXPAND(n)→{m_i}, ADD(n, CLOSED);
(10) SORT(OPEN);
(11) Go to Loop.

4 Simulation Results

One example of a MOTOMAN-K30S robot manipulator motion planning is given here, the simulation runs on a Silicon Graphics 4D/70 workstation(12.5 MHZ, 10 MIPS). Fig. 3 shows the robots motion in an environment with three obstacles, two placed on the ground and one suspended in the air, Fig. 3(a) and Fig. 3(f) are start pose and goal pose respectively. Where, the MaxStep is 16 (°), BasicStep=1.0 (°), w=0.5 in the evaluation function, the planning time for generating the collision-free path is 25 s.

5 Conclusion

The C-space modeling method based on CCJA can compute C-boundary rapidly. The approximation of obstacles with tori allows us to solve 3D collision-free problem in two planes: H-plane and V-plane. With a modified *A** algorithm, the path search has a tremendous speed increase. Simulation results shows that RIPPS is efficient for robot manipulators collision-free path planning in a structured and known environment.

References

1 Lozano Perez. IEEE J of Robot and Automation, 1987, RA-3: 224
2 G Gini, R Massa, R Negretti. J of Robotic System, 1995, 12(2): 93
3 P K Pal, K Jayarajan. In: Proc of IEEE Int Conf on Robotics and Automation. Atlanta: Georgi IEEE Computer Society Press, 1993. 669
4 J H Chuang, N Ahuja. In: Proc of IEEE Int. Conf on Robotics and Automation. San Diego: California IEEE Computer Society Press, 1991. 558
5 Z Shiller, S Dubowsky. Int J of Robotics Research, 1989, 8(6): 3
6 Haizhou Ai, Bo Zhang. Robot (in Chinese), 1994, 16(2): 87
7 J Pearl. Heuristic: Intelligent Search Strategies for Computer Problem Solving. Addison WesleyPress, 1985
8 Wei Wang. Collision-Free Autonomous Path Planning for Robots in Obstacle Environment. (in Chinese): [dissertation]. Harbin: Harbin Institute of Technology, 1997