Robotica (1992) volume 10, pp 321–328

A geometric approach to manipulator path planning in 3D space in the presence of obstacles

Manja Kirćanski and Olga Timčenko

"Mihajlo Pupin" Institute, POB 15, Volgina 15, Belgrade (Yugoslavia)

(Received in Final Form: September 26, 1991)

SUMMARY

The paper presents a geometric method for collision-free manipulator path planning in 3D Euclidean space with polyhedral obstacles. It ensures that none of the links nor the manipulator tip collide with the objects. The method is computationally very cheap and it does not require intensive off-line preprocessing. Hence, it is real-time applicable if the information about obstacles positions and shapes is obtained from a higher control level. The trajectories generated lie within the reachable workspace. The method is implemented on a VAX 11/750 computer and the simulation results are included.

KEYWORDS: Robot; Path planning; Obstacles avoidance; Polyhedral obstacles; Euclidean space; Computational complexity.

1. INTRODUCTION

The improvement of robot autonomy and intelligence is gaining increasing importance in widening the range of applications of robotic devices. Automatic path planning in environments with obstacles, both stationary and moving, considerably increases this ability. This is the reason why path planning for mobile robots (2D) and for classical industrial robots (3D) has achieved considerable attention in several last years.

In this paper we will be mostly concerned with the algorithms dealing with path planning where complete information on the geometry of the robot and the obstacles in the scene is known (either specified in advance or obtained in real-time from higher control levels), but not with the navigation algorithms in unstructured environments that are directly based on the information from sensors.

There are many possible ways to classify all the existing approaches to collision-free path planning. We will consider several distinct categories:

1. Methods based on intensive searches:
   (a) Search in the configuration space[1–5]
   (b) Search in the Cartesian space[6–8]
   (c) Search of Voronoi diagrams[9,10]
2. Papers dealing with optimization of collision-free robot motion considering robots as dynamic systems[11–15]
3. Papers dealing with obstacle avoidance using redundant robots[16–22]
4. Papers dealing with path planning of multi-robot systems[23–27]
5. Heuristic approaches[28–30]
6. Special approaches to obstacle avoidance.[31–33]

Methods based on intensive searches form the most numerous category of collision avoidance methods. The search in the configuration space was first introduced in ref. 1. It was later improved and simplified several times.[2–5] The method provides that a large part of computations is transferred into the off-line preparation (the computation of the configuration space) which is performed only once for the given scene. Search in the Euclidean space directly is proposed too,[6,7] as well as a combination of searches in configuration and Euclidean space.[8] A special group of papers includes the search of Voronoi diagrams[9,10] where unfolding of polyhedral obstacles is being performed. Common features for all these algorithms are the following: They require complex off-line preprocessing for a given scene, and the search of the optimal trajectory lasts very long due to the complexity of the graphs. This implies that the algorithms are not real-time applicable, especially not for moving obstacles. They are practically applicable only for 2D path planning, while in 3D path planning the computational complexity grows exponentially. On the other hand, these methods find the shortest paths in cluttered and rather complex environments.

The algorithms for the generation of optimal collision-free trajectories are based on the minimization of a performance criterion, where the manipulator is modelled by the complete dynamic model and the obstacles are taken into account as constraints on the system state.[11–15] Most usually, the performance criterion is time,[13,14] or energy[11] (in ref. 11 energy is modified by the distance to the obstacles), or directly distance to the obstacles.[15] The minimization is carried out by parameter optimization of trajectories described by spline functions. These methods are off-line in nature, and they require an initial collision-free solution,[11,15] so that they are not real-time applicable. Besides, these algorithms hardly take into account complex environments. The examples are mostly in a 2D space.[11,12,14]

A special method to avoid obstacles is to make use of the surplus of degrees of freedom of redundant manipulators.[16–22] Redundant manipulators offer great possibilities in that regard, since it can be ensured that the manipulator links stay sufficiently far away from the obstacles in 3D space. These algorithms are not off-line in nature, although they require powerful computational resources. A combination of already developed methods

for manipulator tip path planning, together with the algorithms for repelling the manipulator links from the obstacles[16,17,21] might lead to efficient methods for collision-free path planning by means of redundant manipulators for stationary and moving obstacles.

A special class of problems is motion planning of multirobot systems.[23,27] Robots represent a special type of moving obstacles, since their motion is controllable, the movements are very fast and they might depend on the information from sensors. For the latter reason on-line path planning for multirobot systems is desirable. However, this is still a very complex problem. Most of the methods proposed so far deal with 2D path planning of two robots. On the other hand, this problem becomes important for industrial practice in order to minimize the execution time for the tasks when robots work in cooperation.

Papers that could conditionally be categorized as heuristic ones tend to develop relatively simple algorithms. They do not generate strictly optimal trajectories, but offer efficient methods for finding suboptimal trajectories that are also quite satisfactory. Therefore, these algorithms may be directly real-time applicable even for moving obstacles.[28–30]

The final group of papers – papers with special approaches – involves the papers that do not belong to any of the preceding groups and have an original approach to the problem solution.[31–33] In refs. 31 and 33 the obstacles are described by potential functions and the robot is repelled from the obstacle by introducing a force acting to some point on the arm. The force is proportional to the gradient of the potential function.

As we can see, one of the major problems with collision-free path planning for robots is the amount of complexity of computation required. Thus, most of the papers treat 2D path planning problems and require complex off-line preprocessing thus preventing their application to real-time planning in environments with moving obstacles. One way to overcome the problem of complexity is to give up strict optimality and thus reduce computational requirements.

In this paper we propose a simple geometric method for 3D collision-free path planning in an environment with known polyhedral obstacles – both convex and concave. This work was partially inspired by the method of 2D path planning for mobile robots suggested in ref. 30. The information about the obstacles locations and shapes may be either specified in advance or obtained from higher control levels and sensors in real-time. The algorithm does not require complex off-line precomputation. It generates several collision-free trajectories for the manipulator tip and selects the minimal distance, one which surely lies within the reachable workspace and is collision-free not only for the manipulator tip but also for the manipulator links. The motion along the paths is jerkless and smooth. The trajectories obtained are not the absolute minimum ones, but they are quite reasonable and not too roundabout. The algorithm is primarily aimed at nonredundant robots, but a modification of the algorithm for redundant robots based on ref. 17 is possible, too. The efficiency of the algorithm is checked by several examples of an UMS7 manipulator moving between polyhedral obstacles.

2. PROBLEM STATEMENT

Let us consider an n degree-of-freedom manipulator in a safe initial position $q_0 \in Q \subset R^n$, $x_{e0} = f(q) \in X_e \subset R^m$, where q is the vector of joint coordinates $q = [q_1 \cdots q_n]^T$, Q – the n-dimensional space of joint coordinates (configuration space), x_e – the vector of world coordinates describing end-effector position and orientation (completely or partially) in the base reference frame $x_{e0} = [x_{e1} \cdots x_{em}]^T$, X_e – the m-dimensional space of world coordinates (reachable workspace), $f(q)$ – the nonlinear vector function relating between joint and world coordinates. Let us assume that in the manipulator environment there exists a set of k obstacles in the form of polyhedra (both convex and concave) which are specified by the position of their vertices in the base reference frame. The final position of the end-effector $x_{eF} \in X_e$ which the manipulator should reach is specified, too; it is assumed to be safe.

The problem is to determine such a trajectory $x_e(t)$, $t \in [0, T]$ for the manipulator end-effector and the corresponding variation of joint coordinates $q(t)$, so that the manipulator moves from the initial point x_{e0} to the final point x_{eF}, the manipulator and all the links stay far enough from the obstacles, the motion is smooth and jerkless and the path lies within the reachable workspace. At the same time, the trajectory should be as less roundabout as possible.

3. PATH PLANNING STRATEGY

The main idea of the algorithm is to move the manipulator tip along the straight line towards the final position until the manipulator doesn't enter its "visible region". The manipulator visible region is defined as the zone in front of the gripper where the distance between the manipulator tip and the obstacle (measured in the direction of motion) is less than l-tiple length of the terminal link (in this paper $l = 3$ was adopted). At that moment the computation of a deviation point towards which the manipulator tip should turn, should start. Prior to explaining the algorithm itself let us introduce another important term – "critical distance". The critical distance d_c is the distance between the manipulator tip and the obstacle after which the deviation towards a new point must either start, or the motion has to be aborted if a collision-free trajectory was not found. The computation of a deviation point must be completed during the motion within the visible region, before reaching the critical distance. In this paper the critical distance was adopted to be equal to the length of the terminal link.

The computation of the deviation points is relatively simple. Several deviation points are generated around the first obstacle on the straight line between the manipulator tip and the goal position, and then they serve as new initial points. Lastly, a search for the minimum distance collision-free trajectory is performed.

3.1. Convex polyhedral obstacles

We will now describe the algorithm in detail. The notation introduced refers to Figure 1. We will first describe the algorithm for convex polyhedra and then extend it to cover concave obstacles.

(i) First, one should check whether the straight line X_0X_F intersects any of the obstacles. If the path is generated in real-time this information is obtained from sensors. Then, if there exists an obstacle on the path, the polyhedron facet that is first intersected by the trajectory X_0X_F should be determined, as well as the intersection point and the critical point X_c that belongs to the line X_0X_F at the critical distance d_c from the facet.

(ii) Now, potential deviation points (subgoal positions) corresponding to the determined intersection facet are to be computed. We determine one deviation point for each polyhedron edge belonging to the intersection facet, but only for those edges where the intersection point of the common normal between the line X_0X_F and the edge lies on the obstacle (but not out of it). For example, in Figure 1 we look for common normal between X_0X_F and the edges AB, BC, CD and DA. Let us denote them by T_1S_1, T_2S_2, T_3S_3 and T_4S_4. Point S_3 is not taken into further consideration since it does not belong to the body. Points S_1, S_2 and S_4 are used in the same way to obtain potential deviation points X_1, X_2, X_4. The deviation point X_1 is selected to lie in the plane of symmetry (plane β) between facets $ABCD$ and $ABEF$, on the line orthogonal to the edge AB in point S_1. The distance S_1X_1 is chosen to be equal to the critical distance d_c. All the potential deviation points $X_1, \ldots, X_k$ determined in this way are stored for further computations.

(iii) Then one should check whether the knot X_i, $i = 1, \ldots, k$ lies within the robot workspace, and if it does, whether the robot link positions are safe, i.e. none of the links collides with obstacles (the method of checking the collisions between the links and the obstacles will be described later). If the positions of all the links are safe when the end-effector is positioned in X_i, the knot X_i is taken as one of the possible deviation points. The same check might be performed for several points on the path X_cX_i, but this would increase computational demands. Otherwise, the check should be performed later when the complete path is determined. If the workspace is tightly cluttered with obstacles this check would result in rejecting most of deviation points, so in that case it should be performed while this stage of trajectory planning is performed.

(iv) Each deviation point $X_1, \ldots, X_j$, $i = 1, \ldots, k$ (determined in the previous step) is used as an initial node of a new trajectory X_iX_F and the new trajectory is treated in the same way as the previous one X_0X_F (steps 1, 2 and 3 are repeated). The only difference is that the computation of a new critical point X_c is no longer necessary, since the motion will be performed from the point X_i, $i \in \{1, \ldots, k\}$ directly towards the next stage deviation points X_{ij}, $j = 1, 2, \ldots$. All the deviation points determined in this way are stored.

(v) The step (iv) is repeated p times before the search between all the possible knots is started. In our examples we chose $p = 3$. Step (iv) is not repeated exactly p times only if it happens that at some previous level the straight lines from all the deviation points towards the final knot X_F are no more obstructed. The set of the knots the manipulator tip may pass through is presented in the form of a graph in Figure 2.

(vi) At the next step, a search between all the possible trajectories found in the previous five steps is performed according to the shortest path criterion. The search is performed using dynamic programming principles, i.e. starting from the knots determined uttermost, taking into account only those knots whose paths towards the final point X_F are clear (the straight line does not intersect any obstacle). In the case that all the paths are still obstructed, the search for the minimum distance path is performed taking into account all the knots. However, in this case collision-free path planning (steps (i)–(vi)) has

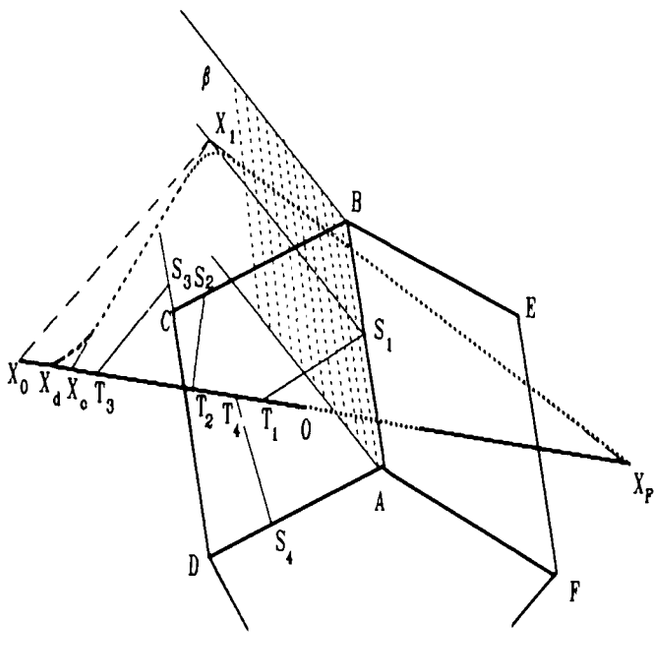

Fig. 1. Illustration of the deviation points calculation.

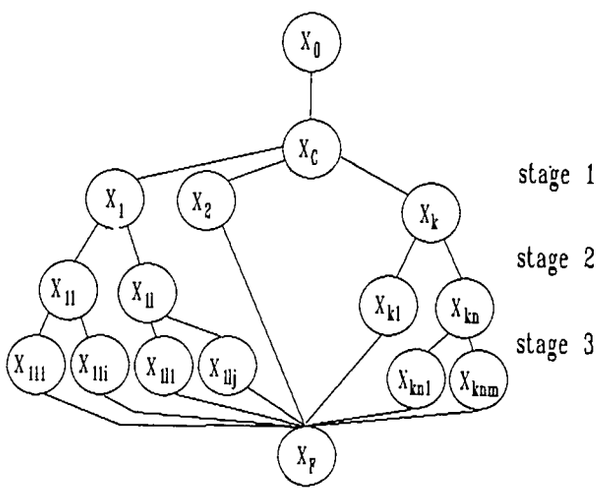

Fig. 2. An example of the deviation points graph.

to be continued during the actual robot motion along the first p trajectory segments.

(vii) Once the minimum distance trajectory is found one should check whether any of the links collides with the obstacles along the whole trajectory (for several points on each trajectory segment, if it has not been checked during the trajectory generation). If the trajectory segment is not collision-free, the next solution with respect to the path distance is to be checked, and so on. If none of the paths that lead directly to the final point X_F was collision-free for the links as well, step (vi) should be repeated for the paths that do not directly lead to the point X_F (there are still some obstacles after p steps of path planning). Then, the minimum distance trajectory should be examined whether it is collision-free for the links, too. If all of these paths are not safe, then the solution to the path planning problem was not found and the robot should be stopped before reaching the critical point X_c. If the safe path was found the path planning will have to be continued during the robot motion (step ix).

(viii) Once the optimal set of deviation points the end-effector should pass through is found, the knot X_d at which the manipulator tip should turn towards the first deviation point has to be determined (Fig. 1). This point belongs to the line X_0X_F and it is relatively close to the robot position when the path planning is completed, but far enough so that smooth transition to a new trajectory segment can be performed.

(ix) In order to provide for continuity of positions, velocities and accelerations the method of continuous path motion proposed in ref. 34 is applied (Fig. 1). The fact that the manipulator tip passes through the neighborhood of the knots (but not exactly through them) is not a drawback here, since the knots themselves are computed heuristically. On the other hand, jerkless motion is very important.

In the case when the straight line between the initial and the final position does not intersect any obstacle, but during the motion some of the links collides with an obstacle, a somewhat modified algorithm may be applied. Namely, the point on the robot that touches the obstacle can be determined, and the above algorithm can be applied to the line between this point and the position of the same point in the final configuration (when the tip is in X_F) instead of applying it to the line X_0X_F.

The path planning algorithm described above generates only positions of the manipulator tip at the deviation point without the gripper orientation management. However, the manipulation task may require the change of orientation specified by the lower part of the initial and final world coordinates vectors x_{e0} and x_{eF}. In this paper, we propose to change the gripper orientation proportionally to the distance travelled, measured along the straight line X_0X_F (the distance between X_0 and the intersection point of the normal from a deviation knot to the line X_0X_F). The motion execution time is evaluated from the specified speed factor. Once the time history of the world coordinates vector $x_e(t)$ is known, the corresponding joint coordinates $q(t)$ are readily obtained by solving the inverse kinematic problem.

One could see in the algorithm described above that the process of generating the deviation knots could be terminated at the stage when the straight line from at least one deviation point to the final point X_F becomes clear, instead of repeating the knots generation p times (e.g. in Figure 1 this would happen at the first stage). This would speed up the path planning process a great deal. However, simulation results showed that such trajectories are usually quite roundabout, and that the search among the knots generated at $p = 3$ stages yielded much better results.

How high the number of stages p at which the computation of deviation knots is performed should be? The answer to this question depends on the complexity of the specific environment and the computational power of the control system. By dividing the trajectory optimization process into several independent parts (if there are many obstacles on the direction X_0X_F) the computational burden is reduced at great deal on account on trajectory optimality in order to achieve real-time applicability. Besides, if the obstacles are moving, it is more convenient to take into account their position immediately before approaching them, instead of using this information much earlier.

Checking whether any of robot collides with any of the obstacles is performed using the following method. The links are considered as bars, while dimensions of obstacles are enlarged to cover the difference between real dimensions of the links and the bars. The possibility of collisions is not absolutely checked (it is checked only in several discrete points). However, by selecting appropriate critical distance d_c the possibility of damages can be practically eliminated.

3.2. Concave polyhedral obstacles

The path planning strategy described above is primarily aimed at convex polyhedra, but it works in many cases of concave bodies too, such as the examples in Figure 3. However, in order to cover the general case of concave polyhedral obstacles the algorithm has to be broadened.

Thus the method of computing the deviation points has to be modified only if the following situation occurs: The straight lines from all the deviation points (computed in the original way) to the final point X_F intersect the same polyhedral facet α as the line X_0X_F did, as illustrated in Figure 4. In other words, in that case an obstacle has a shape similar to a box, and the manipulator's tip trajectory intersects the bottom of the box. This would lead to a hopeless situation. Therefore, a new plane has to be found to repel the manipulator tip properly. Herein we propose the following strategy: First, all the polyhedron facets β_i that touch the facet α are determined, as well as those edges a_i of the facets β_i that do not belong to any other facet β_i (except one) nor the facet α. The edges a_i form a polygon b, but it does not necessarily lie in a single plane. The polygon is divided into triangles by taking 3 by 3 vertices of it (this

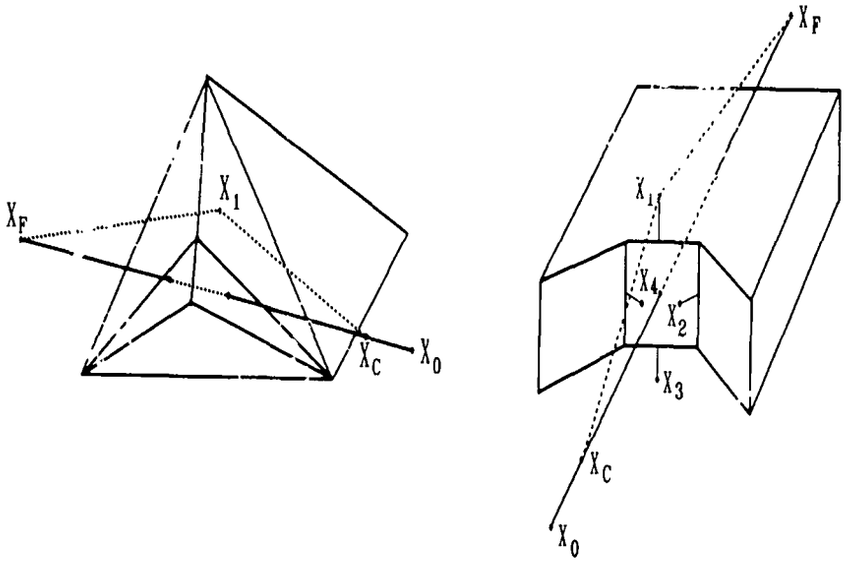

Fig. 3. Examples of concave obstacles where the proposed method works without modifications.

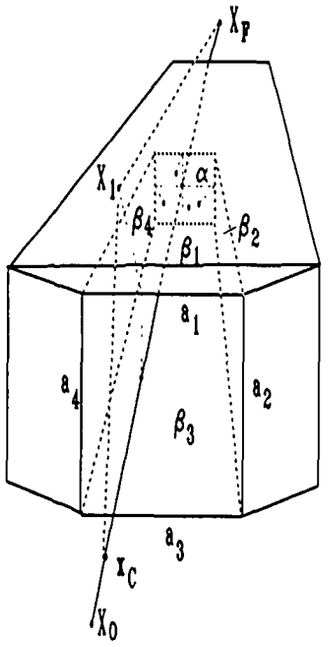

Fig. 4. Illustration of the modified method.

is not unique). The common normals between the line X_0X_F and all the polygon edges are determined, as well as the deviation points for all the polygon edges (arranging for them to lie in the planes of symmetry between the triangles and the polyhedron facets). The rest of the algorithm is the same as before, but the length of the manipulator's visible region, where the path planning process starts, must be proportional to the depth of the obstacle's recesses (if it is greater than the end-effector length).

4. NUMERICAL COMPLEXITY

The proposed algorithm is computationally relatively cheap. For example, the determination whether the path x_0x_F intersects an obstacle requires about 15 pairs of multiplications and additions per each polyhedron facet. The evaluation of a single deviation point requires about 44 pairs of multiplications and additions. For a parallelepiped this would yield a maximum of 1032 pairs of multiplications and additions and 24 square roots. However, since during the evaluation of the deviation points some of them are rejected, this number is reduced for about 50%. The process of searching for the optimal trajectory does not consume significant processor time since the total number of points is relatively small.

The check whether a point lies within the manipulator's workspace takes less than 100 pairs of multiplication and additions (an inverse kinematics computation if the manipulator has a geometrical solution).

The most time-consuming part of the algorithm is checking whether any of robot links collides with any of the obstacles for the desired trajectory. The exact evaluation of the intersection between a bar and a polyhedron requires about 15 pairs of multiplications and additions for each edge of a polyhedron facet maximum. It is repeated for each polyhedron facet maximum. If we have some sensors in the system we can obtain information which obstacles are near a manipulator segment from a higher control level. Then we can check for collision only for some manipulator's segments with those obstacles.

In the case of concave obstacles of the box shape, as it was proposed earlier, the algorithm needs some additional computation. First of all, all the deviation points corresponding to the plane α should be calculated, and then we can determine if the case of box-shaped obstacle has occurred. If polygon b has n_b vertices, we can divide it into $n_b - 2$ triangles with diagonals from a chosen vertex. We have to determine the orths for all the planes that the determined triangles belong to. That

would require 11 additions and 9 multiplications per plane, which takes about 1/4 time needed to calculate a single deviation point. Then $n_b - 2$ deviation points should be determined, and the process of trajectory planning should be continued as it has been described for convex obstacles. The number n_b depends on the shape of the obstacle, but for box-like obstacles it is equal to the number of vertices for the facet α. So, in order to repel the manipulator properly, in this case we need a 9/4 longer time for calculating the deviation points at one stage than in the case of convex obstacles. As we have p stages (or, in the case of a complicated environment, several times p stages), we can see that the computational complexity would not increase a great deal for concave obstacles.

The total number of operations depends a great deal on the location of the robot, its position, the path itself, the type of the objects (number of facets) and so on. In our example in Figure 5 the check whether the links collide with the objects for all possible paths took about 85 pairs of multiplications and additions.

Having in mind that the analytical inverse kinematic solutions require 70–100 multiplications and additions, we can see that we can achieve real-time collision-free path planning by introducing a single processor parallel to the kinematic one with similar computational power. In that case, while robot is moving within the obstacle's visible region, the processor will be able to determine a new collision-free path within 15–20 integration intervals.

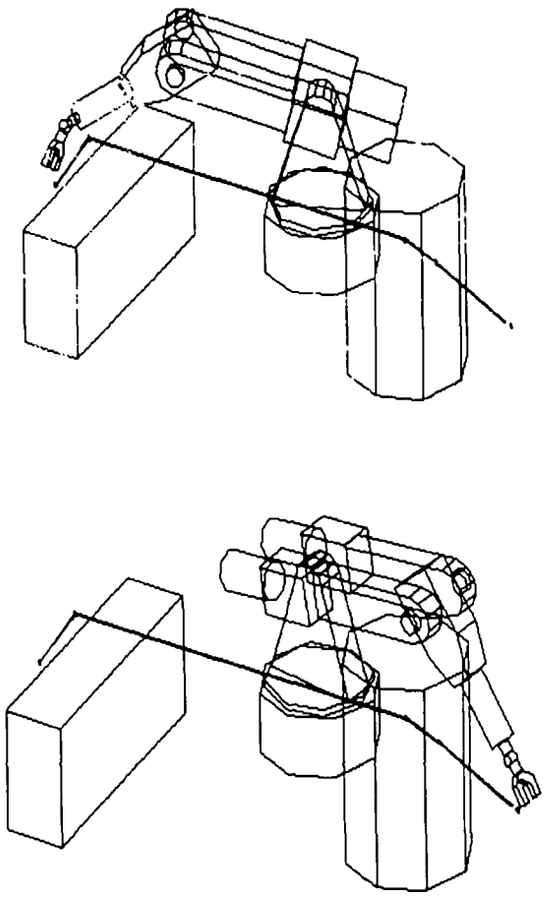

Fig. 5. UMS-7 robot moving among obstacles.

5. EXAMPLES

We will give two examples here.

In our first example we considered the 6 degree-of-freedom manipultor UMS7 moving amongst two objects – a parallelepiped and a prism (Figure 5).

The robot had to move from

$$x_0 = [-1.3, -1.6, 0.6]^T \text{ [m]}$$

to

$$x_F = [0.8, 1.8, 1]^T \text{[m]}$$

with respect to the base frame located at the manipulator base. The algorithm found that the shortest collision-free path passed through points

$$x_3 = [-0.503, -1.446, 1.172]^T \text{ [m]},$$

$$x_{31} = [1.029, 1.476, 1.153]^T \text{ [m]}$$

and

$$x_{312} = [0.938, 1.531, 1.076]^T \text{ [m]}.$$

At the first stage the algorithm found 4 deviation points, and at the second stage it found 3, 2, 2, and 3 deviation points. Two of them where rejected since the links collided with the objects in those postures. At the third stage the algorithm found 0, 2, 3, 0, 2, 3, 2, 3, 2 and 0 deviation points. Among them a minimal collision-free trajectory was chosen and shown in Fig. 5.

In Figure 6 an example of robot going out of a box-like obstacle is presented.

6. DISCUSSION

The method proposed herein is not convenient for tightly cluttered environments. This is due to the fact that the computation of the deviation points itself does not take into account the possible presence of other obstacles in the neighborhood, but the check of collisions with all the bodies is performed later. But in the environments similar to the ones in the examples, the proposed method is very efficient for finding satisfactory, not too roundabout collision-free trajectories. The three levels of trajectory planning described above were quite sufficient for optimizing the trajectory up to the end point. The search for the minimum distance trajectory through the knots graph takes an almost negligible time, since the number of knots is relatively low.

This collision-free path planning method is computationally the cheapest method compared to other methods for 3D path planning proposed so far for single-arm, nonredundant robots.[2,4,6–11,13,14] This feature enables its application to moving polyhedral obstacles that move slowly with respect to the manipulator itself. A higher control level provides the information about the instantaneous obstacles positions by employing a vision system. Since the proposed method leaves in reserve a certain distance between the manipulator and objects, this distance will not change significantly during the task execution if the obstacles are moving much more slowly than the robot. By adjusting the value of the critical

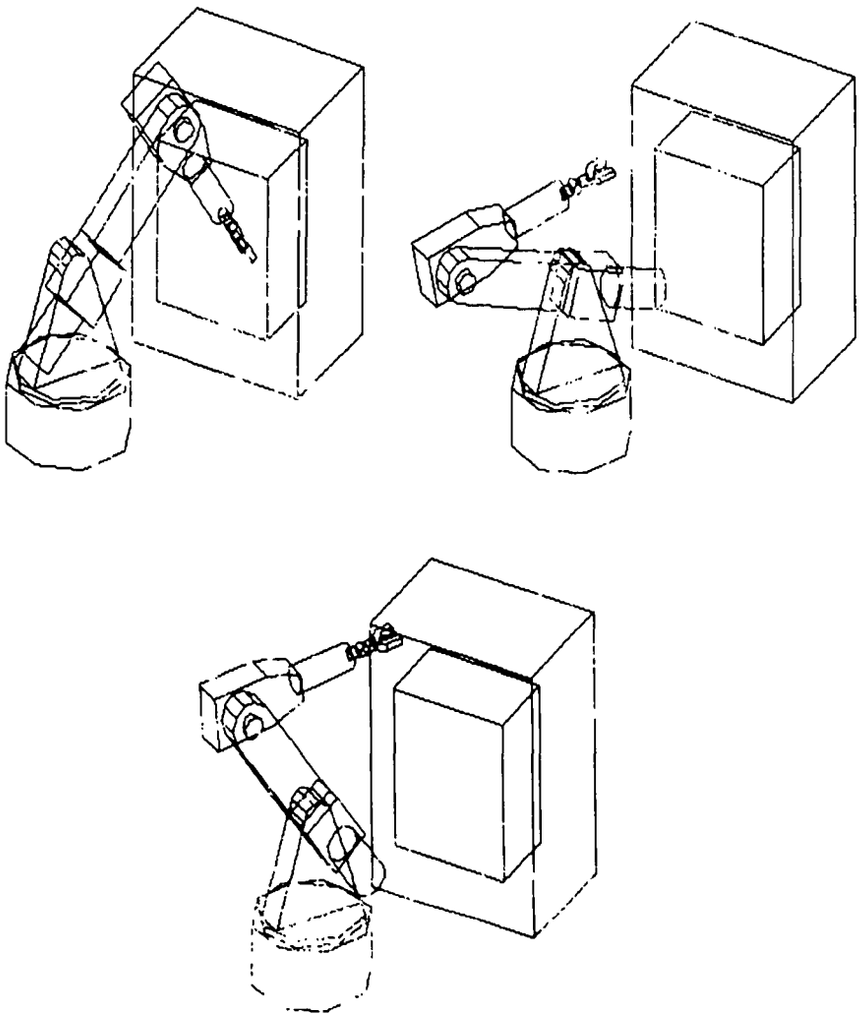

Fig. 6. Initial position of the robot inside a box-like obstacle. The algorithm finds the way out.

distance d_c according to the average speed of obstacles it is possible to achieve sufficient security. Also, by adjusting d_c the problems occurring if the information about the obstacles positions (obtained from a higher control level) is not precise enough, can be lessened. Of course, the selection of a too long critical distance would prevent the finding of a collision-free trajectory although it might exist. At last, the possibility of damage is completely avoided by mounting several proximity sensors on the manipulator links.[28]

7. CONCLUSION

The proposed method solves a very complex problem of 3D planning for a robot in an environment with obstacles. It does not require complex precomputations, thus being convenient for real-time path planning. The positions of the obstacles have to be specified in advance or delivered by a higher control level in real-time. The method works both for stationary and slowly moving obstacles (comparing to the manipulator speed). The low computational complexity was achieved by giving up strict optimality condition. The obtained near-minimum trajectories are quire acceptable and not too roundabout. They surely lie within the reachable manipulator workspace. The robot motion is smooth and jerkless. The method is very efficient in finding collision-free trajectories in environments with several polyhedral obstacles.

References

1. T. Lozano-Perez, "Automatic Planning of Manipulator Transfer Movements" *IEEE Transactions on Systems, Man, and Cybernetics* **11**(10), 681–698 (October, 1981).
2. T. Lozano-Perez, "A Simple Motion-Planning Algorithm for General Robot Manipulators" *IEEE J. of Robotics and Automation* **3**(3), 224–238 (June, 1987).
3. D.T. Kuan, J.C. Zamiska and R.A. Brooks, "Natural Decomposition of Free Space for Path Planning" *Proc. of IEEE Int. Conference on Robotics and Automation*, St. Louis, 168–173 (1985).
4. E.K. Wong and K.S. Fu, "A Hierarchical Orthogonal Space Approach to Three-Dimensional Path Planning" *IEEE J. of Robotics and Automation* **2**(1), 506–513 (March, 1986).
5. D. Zhu and J. C. Latombe, "Constraint Reformulation in a Hierarchical Path Planner" *Proc. of IEEE Int.*

Conference on Robotics and Automation, Cincinnati, 1918–1923 (1990).
6. S. Jun and J.G. Shin, "A Probabilistic Approach to Collision-Free Robot Path Planning" *Proc. of IEEE Int. Conf. on Robotics and Automation,* Philadelphia, 220–225 (1988).
7. J. Ashton and D. Hoang, "An Algorithm for Finding Optimal Paths Between Points While Avoiding Polyhedral Obstacles" *Proc. of Int. Conf. on Advanced Robotics ICAR'87,* Versailles, 307–312 (1987).
8. T. Hasegawa and H. Terasaki, "Collision Avoidance: Divide-and-Conquer Approach by Determining Intermediate Goals" *Proc. of Int. Conf. on Advanced Robotics ICAR'87,* Versailles, 295–306 (1987).
9. C. Bajaj and T.T. Moh, "Generalized Unfoldings for Shortest Paths" *Int. J. of Robotics Research* **7**(1), 71–76 (Feb., 1988).
10. R. Franklin and V. Akman, "Locus Techniques for Shortest Path Problems in Robotics" *Proc. of SyRoCo,* Barcelona, 131–136 (1985).
11. E. Gilbert and D. Johnson, "Distance Functions and their Application to Robot Path Planning in the Presence of Obstacles" *IEEE J. of Robotics and Automation* **1**(1), 21–30 (March 1985).
12. Y. Ch. Chen and M. Vidyasagar, "Optimal Trajectory Planning for Planar n-Link Revolute Manipulators in the Presence of Obstacles" *Proc. of IEEE Conf. on Robotics and Automation,* Philadelphia, 202–208 (1988).
13. Z. Shiller and S. Dubowsky, "Global Time Optimal Motions of Robotic Manipulators in the Presence of Obstacles" *Proc. of IEEE Conf. on Robotics and Automation,* Philadelphia, 370–375 (1988).
14. M. Takano and K. Sasaki, "Time Optimal Control of PTP Motion of a Robot with Collision Avoidance" *Proc. of Int. Conf on Advanced Robotics ICAR'87,* Versailles, France, 445–456 (1987).
15. R. O. Buchal and D. B. Cherchas, "An Iterative Method for Generating Kinematically Feasible Interference-Free Robot Trajectories" *Robotica* **7**(2), 119–127 (1989).
16. A. Maciejewski and Ch. Klein, "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments" *Int. J. Robotic Research* **4**(3), 109–117 (1985).
17. M. Kirćanski and M. Vukobratović, "Contribution to control of Redundant Robotic Manipulators in an Environment with Obstacles" *Int. J. Robotic Research* **5**(4), 112–119 (Winter 1987).
18. V.L. Generozov, "An Algorithm for Manipulator Trajectory Planning in the Presence of Obstacles" *Technicheskaya Kibernetika* **1,** 137–147 (1984).
19. G. Aksenov, D. Voroneckaya and V. Fomin, "Generation of Programmed Manipulator Motions by Means of Computers" *Technicheskaya Kibernetika* **4,** 50–55 (1978).
20. J. Baillieul, "Avoiding Obstacles and Resolving Kinematic Redundancy" *Proc. of IEEE Conf. on Robotics and Automation,* San Francisco, 1698–1704 (1986).
21. B. Faverjon and P. Tournassoud, "A Local Based Approach for Path Planning of Manipulators with a High Number of Degrees of Freedom" *Proc. of IEEE Conf. on Robotics and Automation,* Raileigh, 1152–1159 (1987).
22. G. S. Chirikjian and J. W. Burdick, "An Obstacle Avoidance Algorithm for Hyper-Redundant Manipulators" *Proc. of IEEE Conf. on Robotics and Automation,* Cincinnati, 625–631 (1990).
23. E. Freund and H. Hoyer, "Real-Time Pathfinding in Multirobot Systems Including Obstacle Avoidance" *Int. J. Robotic Research* **7**(1), 42–70 (February, 1988).
24. B.H. Lee and C.S.G. Lee, "Collision-Free Motion Planning of Two Robots" *IEEE Trans. on Syst., Man, and Cyber.* **17**(1), 21–32 (Jan./Feb., 1987).
25. J. Roach and M. Boaz, "Coordinating the Motions of Robot Arms in a Common Workspace" *IEEE J. of Robotics and Automation* **3**(5), 437–444 (Oct., 1987).
26. P. Tournassoud and B. Faverjon, "Cooperation of Two Manipulators" *Proc. of Int. Conf. on Advanced Robotics ICAR'87,* Versailles, France, 313–324 (1987).
27. T. Nagata, K. Honda and Y. Teramoto, "Multirobot Plan Generation in a Continuous Domain: Planning by Use of Plan Graph and Avoiding Collision Among Robots" *IEEE J. of Robotics and Automation* **4**(1) (February, 1988).
28. Y. Yamada, K. Iwata, H. Yonekura, H. Seiki, N. Tsuchida and M. Ueda, "Collision-Free Control of a 3-Link Arm by Using Ultrasonic Proximity Sensors" *Proc. of 15th ISIR,* Tokyo, 943–952 (1985).
29. C. Ramirez, "Stratified Levels of Risk for Collision-Free Robot Guidance" *Proc. of 15th ISIR,* Tokyo, 959–966 (1985).
30. Y. Ichikawa and N. Ozaki, "Autonomous Mobile robot", *J. of Robotic Systems,* **2**(1), 135–144 (1985).
31. O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots" *Int. J. of Robotic Research* **5**(1), 90–99 (Spring 1986).
32. J. F. Canny and M. C. Lin, "An Opportunistic Global Path Planner" *Proc. of IEEE Conf. on Robotics and Automation,* Cincinnati, 1554–1559 (1990).
33. V. Lumelsky, "Effect of Kinematics and Motion Planning for Planar Robot Arms Moving Amidst Unknown Obstacles" *IEEE J. of Robotics and Automation* **3**(3), 207–223 (June, 1987).
34. R. Paul, *Robot Manipulators: Mathematics, Programming, and Control* (The MIT Press, Cambridge, Massachusetts, 1981).