Obstacle Avoidance for Manipulators

Wei Zhang and Tarek M. Sobh

Department of Computer Science and Engineering
University of Bridgeport
Bridgeport, CT 06601, USA

Abstract

This paper presents an obstacle avoidance approach for manipulators based on an obstacle avoidance path planning mechanism. Although the obstacle dealt with in the paper is of the cubic type, it can be transformed to the spherical type as shown in the paper, which has a simpler analytical description. The obstacle avoidance problem has been formulated in terms of collision avoidance of links rather than points. Link collision avoidance is achieved by continuously controlling the link closest point to the obstacle. For the robot SIR-1 which has an articulated chain, a link can be represented as the line segment defined by the Cartesian position of its two neighboring joints. We implement a simple obstacle avoidance path planning algorithm.

1 Introduction

For the most part, we will consider motions of a manipulator as motions of the tool frame, T, relative to the station frame, S. When paths are specified as motion of the tool frame relative to the station frame, we decouple the motion description from any particular robot, end-effector, or workpieces. This results in a modularity description, and would allow the same path description to be used for a different manipulator, or for the same manipulator with a different tool. Further, we can specify and plan motions relative to a moving work station (e.g., a conveyor belt) by planning motions relative to the station frame. At run time, the definition of S will be time variant.

The basic problem is to move the manipulator from an initial position to some desired final position. That is, we wish to move the tool frame from its current value, T_{init}, to a desired final value, T_{final}. Note that this motion in general involves a change in orientation as well as a change in the position of the tool relative to the station.

For the purpose of obstacle avoidance, the path of the end-effector can be further constrained by the addition of points intermediate to the initial and final configurations. It is necessary to specify the motion in much more detail than simply stating the desired final configuration. One way to include more detail in a path description is to supply a sequence of intermediate points between the initial and final positions. Along with these spatial constraints on the motion, the user may also wish to specify temporal attributes of the motion. We define a smooth function for fitting the trajectory. The function has a linear first derivative. Jerky motions tend to cause increased wear on the mechanism, and cause vibrations by exciting resonance in the manipulator.

2 Path description and generation

For path planning, we are concerned only with position of the end-effector in spatial space, since the orientation of the end-effector will not change the path planned.

In order to make the description of manipulator motion easy to define for the user of a robot system, the user should not be required to write down complicated functions of space and time to specify the task. We allow the capability of specifying paths with simple descriptions of the desired motion, and let the system figure out the details. For example, the user may just specify the desired goal position and orientation of the end-effector, and leave it to the system to decide on the exact shape of the path to get there, the duration, the velocity profile, and other details. Path generation occurs at run time and, in the most general case, position, velocity, and acceleration are computed. Since these paths are computed on digital computers, the path points are computed at certain rate, called the path update rate. In typical manipulator systems this rate ranges between 20 and 200 Hz.

3 Joint space schemes

Here we consider methods of path generation in which the path shapes (in space and in time) are described in terms of functions of joint angles. Each path point is usually specified in terms of a desired position and orientation of the tool frame, T, relative to the station frame, S. Each of these points is "converted" into a set of desired joint angles by the application of the inverse kinematics. Then a smooth function is determined for each of the n joints which pass through the points and end at the goal point. The time required for each segment is the same for each joint so that all joints will reach the goal point. Other than specifying the same duration for each joint, the determination of the desired joint angle function for a particular joint does not depend on the functions for the other joints.

4 Inverse kinematics for the SIR-1 robot manipulator

For the SIR-1 robot manipulator, given the position as $P = x, y, z$; the required first three joint variables are evaluated as follows:

$\theta_1 = \arctan(x, y)$

$\theta_2 = \arctan(\sqrt{x^2 + y^2}, z - d_1) - \arctan(a_2 + a_3 c_3, a_3 s_3)$

$\theta_3 = \arctan(D, \pm\sqrt{1 - D^2})$,

where $D = \frac{x^2+y^2+(z-d_1)^2-a_2^2-a_3^2}{2a_2a_3}$

The motion of the first three joints is calculated by computing the joint variables $\theta 1$, $\theta 2$, and $\theta 3$, corresponding to P. So we can use the above formula to convert any P in Cartesian space to its corresponding Joint space angles.

5 Cubic polynomials for a path with intermediate points

In general, we wish to allow paths to be specified which include intermediate points. If the manipulator is to come to rest at each point, then we can use the cubic formulation [1]. Usually, we wish to be able to pass through an intermediate point without stopping, and we need to generalize the way in which we fit cubics to the path constraints.

As in the case of a single goal point, each point is specified in terms of a desired position and orientation of the tool frame relative to the station frame. Each of these points is converted into a set of desired joint angles by application of the inverse kinematics. We then consider the problem of computing a cubic which connects the point values for each joint smoothly way.

If the desired velocities of the joints at the points are known, then we can determine cubic polynomials using the standard cubic fitting technique, but now the velocity constraints at each end are not zero, but rather, some known velocity. The constraints become:

$\dot{\theta}(0) = \dot{\theta_0}$

$\dot{\theta}(t_f) = \dot{\theta_f}$

The four equations describing this general cubic are:

$\theta_0 = a_0$

$\theta f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3$

$\dot{\theta_0} = a_1$

$\dot{\theta_f} = a_1 + 2a_2 t_f + 3a_3 t_f^2$

Solving these equations for a_i, we obtain:

$a_0 = \theta_0$

$a_1 = \dot{\theta_0}$

$a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0) - \frac{2}{t_f}\dot{\theta_0} - \frac{1}{t_f}\dot{\theta_f}$

$a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0) + \frac{1}{t_f^2}(\dot{\theta_f} - \dot{\theta_0})$

We can then calculate the cubic polynomial that connects any initial and final position and velocities.

6 Collision-free path planning for the SIR-1 robot manipulator

Now, let us consider the obstacle avoidance, or collision-free, path planning for the SIR-1 robot manipulator. It would be extremely convenient if we could simply tell the robot system what the desired goal point of the manipulator motion is, and let the system determine where and how many points are required so that the goal is reached without the manipulator hitting any obstacles. In order to do this, the system must have models of the manipulator, the work area, and all potential obstacles in the area. A second manipulator may even be working in the same area and hence each arm must be considered as a moving obstacle for the other.

The obstacle we will deal with is of a cubic form. For simplifying the problem, we assume that there is only one cube in work area, so the problem becomes how to find a set of points so that the SIR-1 robot manipulator can go from the initial position to the final position without hitting the cube.

Considering the presentation of a cube in Cartesian space, we notice that for any cube in the work area, we can always find a sphere to cover it completely. The mathematical description of the sphere in Cartesian space is much simpler than the cube. In order to avoid hitting an obstacle each intermediate point must reside on or above the surface of the sphere, and the end-effector as well as the links of the robot should not cut through the sphere surface. We can visualize the geometric problem in Figure 1.

7 Robot obstacle avoidance: A geometric approach

The manipulator obstacle avoidance problem has been formulated in terms of collision avoidance of links rather than points. Link collision avoidance is achieved by continuously controlling the link's closest point to the obstacle. Additional links can be artificially introduced or the length of the last link can be extended to account for the manipulator tool or load. For SIR- 1 robot, a link can be represented as the line segment defined by the Cartesian position of its two neighboring joints.

The axes of the frame of reference R are chosen such that its z-axis is the base z-axis of manipulator and its origin is at its bottom. The manipulator and obstacle parameters are designated by $d_1, a_2, a_3, \theta_1, \theta_2, \theta_3, R$ and D , respectively. Figures 2 and 3 depict the geometric link constraints.

First let us take a look down along the z-axis and we can get the obstacle range for θ_1, as follows:

$r^2 = R^2 - D^2 \sin^2(\theta_1)$; $d = D\cos(\theta_1)$.

To guarantee that the link will not to hit the obstacle, D must be greater than R, otherwise the obstacle would be too large to be avoided. The range for θ_1 of the obstacle thus can be expressed as $|\theta_1| \leq \arctan(\sqrt{D^2 - R^2}, R)$.

Then we calculate the distances of link 2 and link 3 to the surface of the sphere. We project the sphere onto the plane which is formed by line L. Thus, we reduce the 3-D problem into a 2-D problem. From the figure we obtain:

$L_2 = \acute{D}\sin(\theta_2 + \alpha)$;

where:

$\alpha = \arctan(d, \acute{d_1})$; $\acute{D} = \sqrt{\acute{d_1^2} + d^2}$;

$L_3 = P\cos(\theta_3 - \beta)$; where $\beta = \arctan(L_2, a_2 - \sqrt{\acute{D}^2 - L_2^2})$;

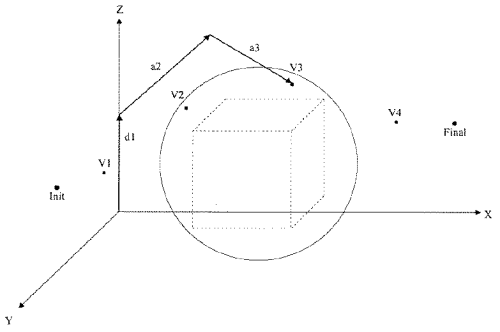

Figure 1: Cubic obstacle and spherical enclosure.

$$P = \sqrt{a_2^2 - 2a_2\sqrt{\acute{D}^2 - L_2^2} + \acute{D}^2};$$

These are distances of link 2 and link 3 to the surface of the sphere.

Now let L_2 and L_3 be equal to r, then we get the minimum θ_2 and θ_3, as follows:

$\theta_{2min} = \arctan(\pm\sqrt{\acute{D}^2 - r^2}, r) - \alpha$

$\theta_{3min} = \arctan(r, \pm\sqrt{P^2 - r^2}) + \beta$

or let r be equal to L_2 or L_3 to get the minimum θ_1 as follows:

$\theta_{1min} = max(\arctan(\sqrt{D^2 + L_2^2 - R^2}, \sqrt{R^2 - L_2^2}),$

$\arctan(\sqrt{D^2 + L_3^2 - R^2}, \sqrt{R^2 - L_3^2}))$

Thus; in order to avoid link collision with an obstacle, θ_1, θ_2 and θ_{31} , must satisfy the above conditions as well as the point avoidance conditions discussed below.

8 Summary and Discussion

We have developed an algorithm for path planning and avoidance of spherical obstacles. Given initial, final and obstacle configurations, first, we convert them into joint space representations, then calculate the path with cubic polynomial functions without the obstacle; then we take the obstacle into consideration: when θ_1 enters the range, that would mean that the manipulator has already hit the object, then we apply the following algorithm to calculate the in-between path; when θ_1 goes out of the range, operation resumes.

Point avoidance occurs when θ_2 and $\theta 3$ do not satisfy the above conditions, i.e., $L_2 < r$ or $L_3 < r$; if this condition happens, the following conditions must be met:

$k_2 = \sqrt{\acute{D}^2 - L_2^2} - \sqrt{r^2 - L_2^2} > a_2$

$k_3 = \sqrt{P^2 - L_3^2} - \sqrt{r^2 - L_3^2} > a_3$

Here, K_2 and K_3 stand for the lengths between start point of a link to the surface of the sphere. Now, let us summarize the algorithms of finding points that are exhibit not only point-avoidance but also link-avoidance.

Algorithm:

1. Calculate inverse kinematics of initial and final points;
2. Decide what mode the algorithm will work on;
3. For each θ_1, check $|\theta_1| \leq \arctan(\sqrt{D^2 - R^2}, R)$, if YES, goto step 4; otherwise continue step 3;
4. For each θ_2, calculate L_2;
5. If $L_2 \geq r$, link 2 does not hit the obstacle (link avoidance condition), goto step 8;
6. If $K_2 > a_2$, link 2 does not hit the obstacle (point avoidance condition), goto step 8;
7. Link 2 hit the obstacle, set $\theta_1 = \theta_{1min}$ for mode 1, or $\theta_2 = \theta_{2min}$ for mode 0;
8. For each θ_3 and L_2, calculate L_3;

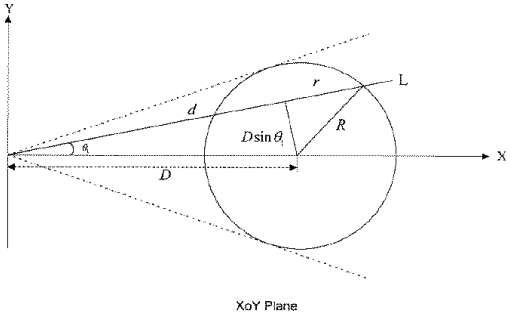

Figure 2: Link intersection with the XY Plane.

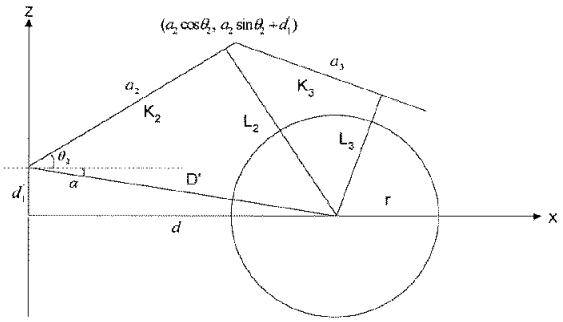

Figure 3: Link intersection with the XZ Plane.

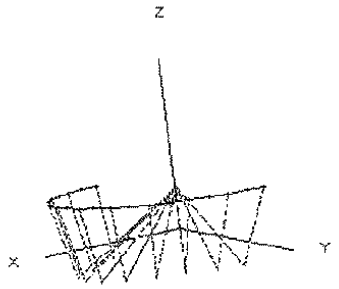

Figure 4: Path 1 without obstacle

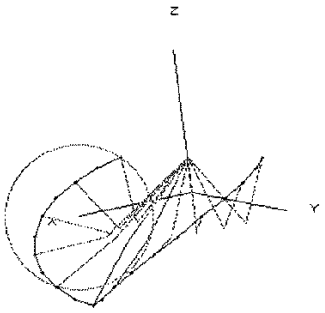

Figure 5: Path 1 with obstacle

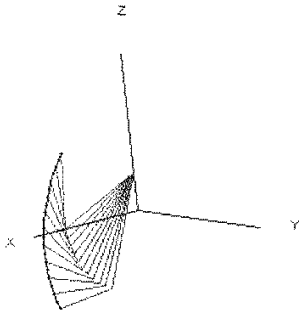

Figure 6: Path 2 without obstacle

9. If $L_3 \geq r$, link 3 does not hit the obstacle (link avoidance condition), goto step 12;
10. If $K_3 > a_3$, link 3 does not hit the obstacle (point avoidance condition), goto step 12;
11. Link 3 hit the obstacle, set $\theta_1 = \theta_{1min}$for mode 1, or $\theta_3 = \theta_{3min}$ for mode 0;
12. Stop. The results will be the set of points represented by joint space angles and can be used for controlling the robot to reach the goal position.

9 Results and Comparisons

Following are some sample runs of the above algorithm. In all the examples, we present the visual description of the path without an obstacle first; and then the modified path in the presence of an obstacle.

9.1 Example 1

In this example, the parameters for the manipulator are: $d_1 = 1$, $a_2 = 3$, $a_3 = 2$. The initial and final points are: $P_0 = (0, 3, 1)$, $P_f = (0, -3, 1)$. The obstacle parameters are $D = 4$, $R = 2$. The figures 4 and 5 depict the original and modified paths.

9.2 Example 2

In this example, the parameters for the manipulator are: $d_1 = 1$, $a_2 = 3$, $a_3 = 2$. The initial and final points are: $P_0 = (3, 0, -2)$, $P_f = (3, 0, 2)$. The obstacle parameters are $D = 4$, $R = 2$. The figures 6 and 7 depict the original and modified paths.

In the results, we can see the differences between the paths with and without the obstacle. The algorithm's simplicity, efficiency, and closed form makes it a viable option to use in real-time robot path generation in the presence of obstacles.

References

[1] Craig, J. *Introduction To Robotics.* Addison-Wesley, 1989.

[2] Khatib, O. Real time obstacle avoidance for manipulators and mobile robots. *Int. J. Robotics Research 3*, (1984)

[3] Lozano-Perez, T. A simple Motion-Planning Algorithm for General Robot Manipulators. *IEEE Journal of Robotics and Automation*, (1987).

[4] Spong, V. *Robot Dynamics and Control.* Wiley, 1989.

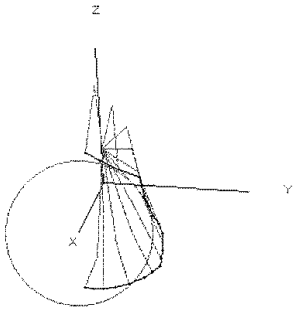

Figure 7: Path 2 with obstacle