7. TRAJECTORY PLANNING

7.1 Introduction

To execute its programmed task the robot must move to a series of locations or “points”. Each point specifies the desired position and orientation of the end-effector. In most applications it is necessary that the robot move in a smooth, controlled manner from one point to the next. A “path” is an ordered sequence of these points. An example of a path is shown in Figure 7.1. A “trajectory” is a path with timing specified. The problem of generating the intermediate points along the trajectory is known as “trajectory planning”. We will study solutions to this problem in this chapter. The methods are similar to those used to generate trajectories for CNC machines (also known as “interpolation”), and those used to design cam profiles.

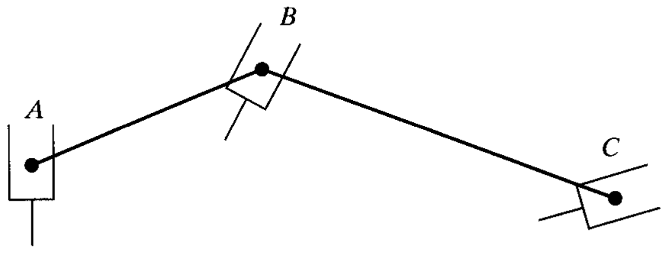

Figure 7.1 Example of a path [1].

7.2 Joint Space vs. Cartesian Space

We may create a trajectory for a robot using two distinct approaches. For example, let’s say we want to move a robot from a start point *A* to an end point *B*. With the first approach, the sequence of movements between *A* and *B* are described in Cartesian space. The intermediate Cartesian points are converted to joint variables (for commanding each of the motors) using the inverse kinematics equations as the robot moves. This is known as “Cartesian space” trajectory planning and has the following advantages:

- The trajectories are easy to visualize since we are familiar with Cartesian space.
- Tasks requiring Cartesian space motions such as moving along a straight line can be easily achieved.

It also has the disadvantages:

- The trajectory may include (or come close to) a singular configuration that leads to dangerously high joint velocities. An example is shown in Figure 7.2.
- Intermediate points could be outside of the robot’s workspace. An example is shown in Figure 7.3.
- The intermediate points may not be feasible without changing the robot’s configuration. An example is shown in Figure 7.4

With the second approach, the sequence of movements between *A* and *B* are described in what is termed “Joint space”. The points A and B are first converted into the equivalent joint values using inverse kinematics. The trajectory generation then involves interpolating between these two sets of joint variables. Joint space trajectory planning avoids all of the disadvantages of Cartesian space trajectory planning. Its only disadvantage is that Cartesian space motions such as straight lines are very difficult to accomplish. For this reason the general rule of thumb when programming robots is to use joint space trajectories by default, and only use Cartesian space trajectories when required.

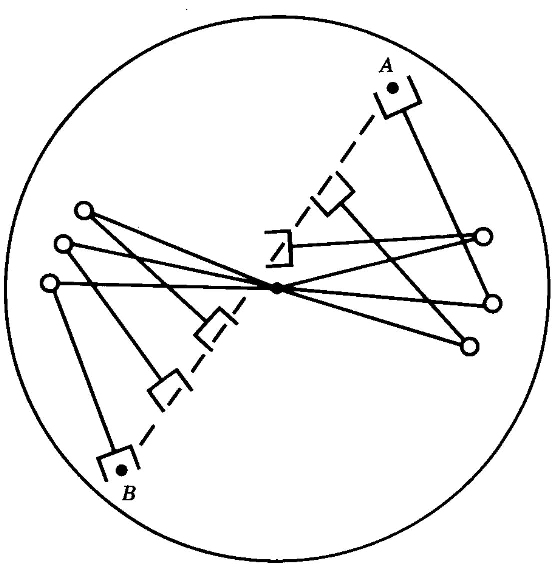

Figure 7.2 A Cartesian path whose intermediate points are close to a singularity [2].

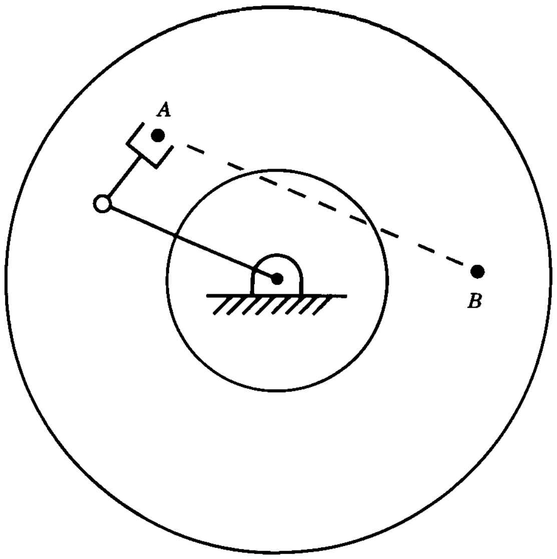

Figure 7.3 A Cartesian path whose intermediate points are outside the workspace [2].

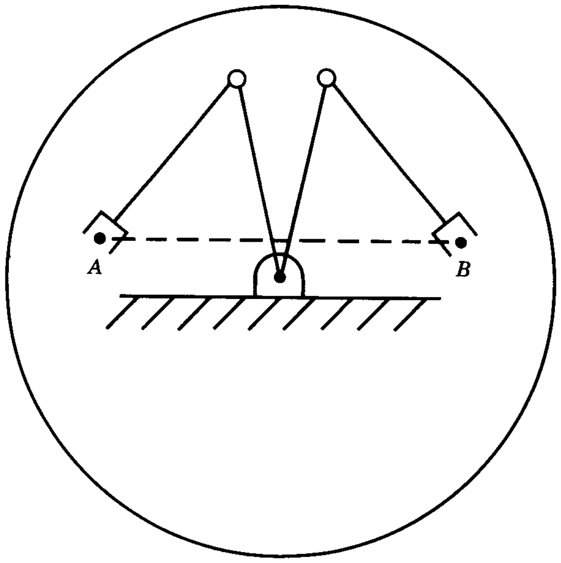

Figure 7.4 A Cartesian path that requires the robot to change configuration.

7.3 Joint Space Trajectory Planning

Various schemes may be used for joint space trajectory planning. The most common schemes are:

1) Using third-order polynomials (or cubic splines).
2) Using fifth-order polynomials (or quintic splines).
3) Using linear segments with parabolic blends (LSPB).

The motion between two points is termed a "segment" and normally a trajectory will include more than one segment. The example shown in Figure 7.1 contains two segments: AB and BC. For the motion to be smooth the segments must be blended together in a controlled way. Using third-order polynomials (scheme1) allows the position and velocity to be made continuous between segments. However the acceleration will not be smooth and may cause the arm to vibrate. A two segment example is shown in Figure 7.5.

Question: When does the acceleration jump in this example?

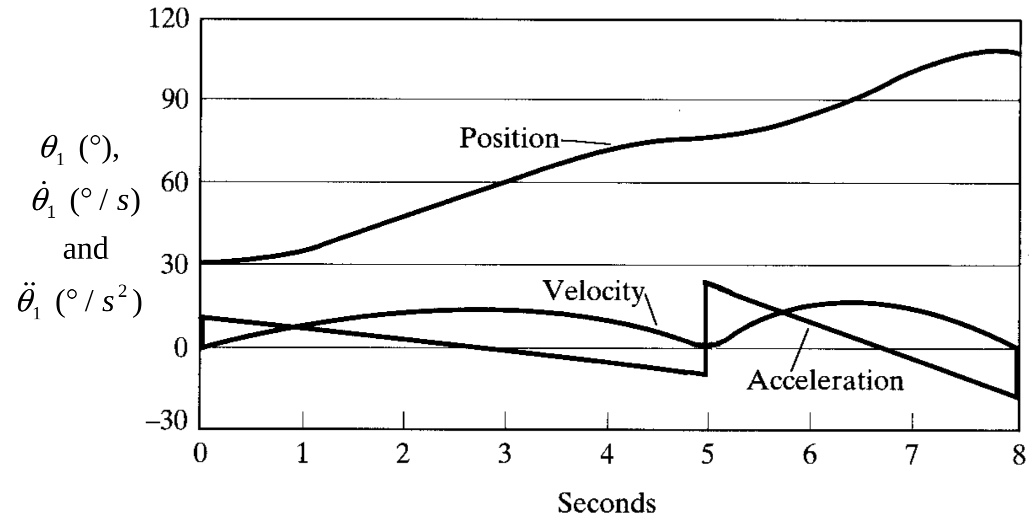

Figure 7.5 Example of a two segment trajectory planned for one joint using third-order polynomials. The first segment ends when t = 5 s. Note: the version of this Figure in Niku [2] is incorrect.

The second scheme using fifth-order polynomials has the advantage that the acceleration can be made smooth throughout the trajectory. It has the disadvantage that the calculations involved are more complex and require more computing speed. The third scheme, LSPB, is the most common and will be the subject of the rest of this chapter.

The simplest choice for trajectory generation would be to linearly interpolate between the joint variables at the start and end of each segment. This is the same as requiring that the joints move at a constant velocity during each segment. (Note that the velocities would have to be chosen so that the joints start and stop at the same time). However this simplistic approach would require infinite accelerations at the start and end of each segment. To avoid the need for infinite accelerations the LSPB scheme blends together the linear portions with parabolic sections. Two potential trajectories for a single segment are shown in Figure 7.6.

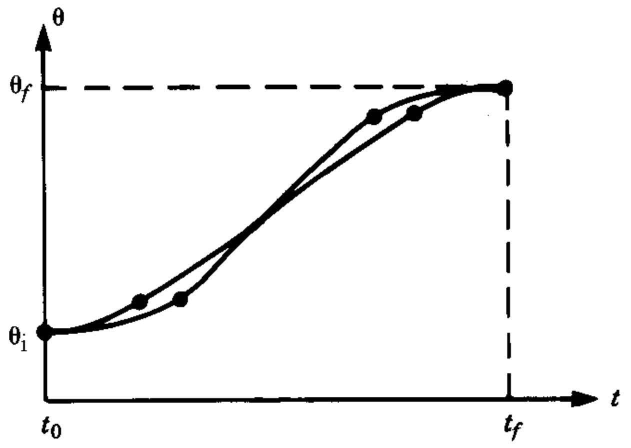

Figure 7.6 Two possible LSPB trajectories for the same movement distance and duration [2].

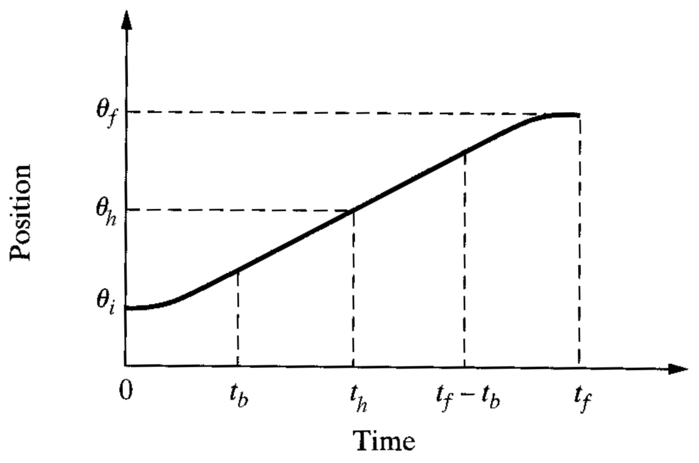

Figure 7.7 Definition of blend and halfway times [1].

Note that the relative duration of the linear and parabolic sections is adjustable. If the durations of the parabolic sections at the start and end of the segment are equal then the corresponding acceleration magnitudes will also be equal. An example is shown in Figure 7.8. The trajectory will also be symmetric about the halfway point where:

$$t = t_h = t_f / 2 \quad \text{and} \tag{7.1}$$

$$\theta = \theta_h = (\theta_i + \theta_f)/2 \tag{7.2}$$

Normally the initial position, θ_i, the desired final position, θ_f, the desired final time, t_f and the desired acceleration $\ddot{\theta}_d$ are known (The first three are determined by the user, the acceleration is usually set within a limit provided by the robot manufacturer). The blend time, t_b, the position at the blend time, θ_b, and the maximum velocity, $\dot{\theta}_{\max}$, are unknown. Please see the plots in figures 7.7 and 7.8. The velocity during the linear section is constant and equals $(\theta_h - \theta_b)/(t_h - t_b)$. This is also the maximum velocity. The velocity at the blend time must equal this velocity so we have:

$$\dot{\theta}_{\max} = \frac{\theta_h - \theta_b}{t_h - t_b} = \ddot{\theta}_d t_b \tag{7.3}$$

At the blend time:

$$\theta_b = \theta_i + \tfrac{1}{2}\ddot{\theta}_d t_b^2 \tag{7.4}$$

Substituting θ_b from (7.4), and $t_f = 2t_h$ from (7.1), into (7.3) gives:

$$\ddot{\theta}_d t_b^2 - \ddot{\theta}_d t_f t_b + (\theta_f - \theta_i) = 0 \tag{7.5}$$

This has the solution:

$$t_b = \frac{t_f}{2} - \frac{\sqrt{\ddot{\theta}_d^2 t_f^2 - 4\ddot{\theta}_d(\theta_f - \theta_i)}}{2\left|\ddot{\theta}_d\right|} \tag{7.6}$$

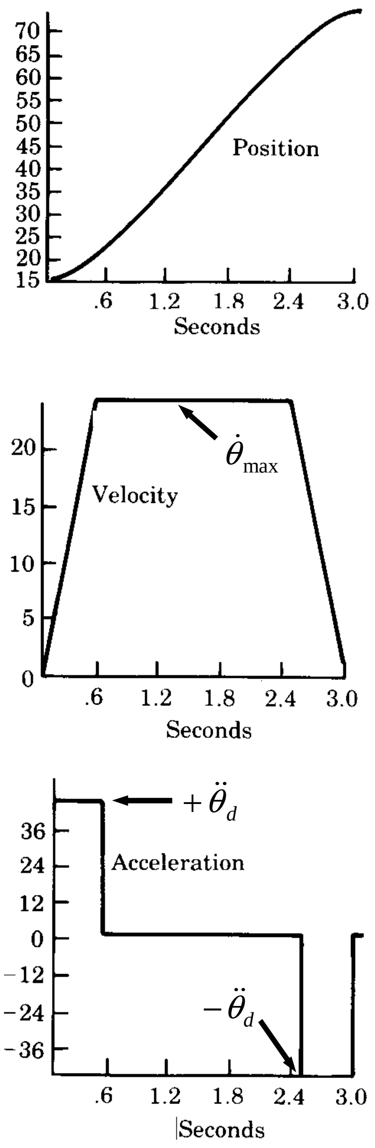

Figure 7.8 Position, velocity and acceleration vs. time for an example LSPB trajectory [2].

Note that when $\theta_f < \theta_i$ the value of $\ddot{\theta}_d$ must be negative. Also note that most textbooks ignore this point and produce wrong answers! With t_b known, the trajectory may be calculated for the three sections as follows:

For $0 \le t \le t_b$:

$$\theta(t) = \theta_i + \tfrac{1}{2}\ddot{\theta}_d t^2,\ \dot{\theta}(t) = \ddot{\theta}_d t, \text{and}\ \ddot{\theta}(t) = \ddot{\theta}_d \tag{7.7}$$

For $t_b < t < (t_f - t_b)$:

$$\theta(t) = \theta_i + \tfrac{1}{2}\ddot{\theta}_d t_b^2 + \ddot{\theta}_d t_b (t - t_b),\ \dot{\theta}(t) = \ddot{\theta}_d t_b, \text{and}\ \ddot{\theta}(t) = 0 \tag{7.8}$$

For $(t_f - t_b) \le t \le t_f$:

$$\theta(t) = \theta_f - \tfrac{1}{2}\ddot{\theta}_d (t_f - t)^2,\ \dot{\theta}(t) = \ddot{\theta}_d (t_f - t), \text{and}\ \ddot{\theta}(t) = -\ddot{\theta}_d \tag{7.9}$$

Example 7.1

We are interested in planning the joint space trajectory for a 2R robot for one motion segment using the LSPB method. The starting point of the segment is $\theta_1(0)=10°, \theta_2(0)=25°$, and the end point is $\theta_1(t_f)=90°$, $\theta_2(t_f)=60°$. The desired acceleration magnitude is $200°/s^2$ and t_f = 1.5 s for both joints, determine t_b and $\dot{\theta}_{\max}$ for each joint, and plot the joint angle trajectories.

For joint 1, $\theta_f > \theta_i$ so we have $\ddot{\theta}_{d1} = 200°/s^2$. Substituting this and the given information into equation (7.6) gives for joint 1:

$$\begin{aligned} t_{b1} &= \frac{t_{f1}}{2} - \frac{\sqrt{\ddot{\theta}_{d1}^{\,2} t_{f1}^{\,2} - 4\ddot{\theta}_{d1}(\theta_{f1}-\theta_{i1})}}{2\left|\ddot{\theta}_{d1}\right|} \\ &= \frac{1.5}{2} - \frac{\sqrt{200^2 1.5^2 - 4(200)(90-10)}}{2(200)} \\ &= 0.347 \text{ s} \end{aligned} \tag{7.10}$$

and

$$\dot{\theta}_{\max 1} = \ddot{\theta}_{d1} t_{b1} = (200)(0.347) = 69.4°/s \tag{7.11}$$

Similarly for joint 2:

$$\begin{aligned} t_{b2} &= \frac{t_{f2}}{2} - \frac{\sqrt{\ddot{\theta}_{d2}^{\,2} t_{f2}^{\,2} - 4\ddot{\theta}_{d2}(\theta_{f2}-\theta_{i2})}}{2\left|\ddot{\theta}_{d2}\right|} \\ &= \frac{1.5}{2} - \frac{\sqrt{200^2 1.5^2 - 4(200)(60-25)}}{2(200)} \\ &= 0.128 \text{ s} \end{aligned} \tag{7.12}$$

and

$$\dot{\theta}_{\max 2} = \ddot{\theta}_{d2} t_{b2} = (200)(0.128) = 25.5°/s \tag{7.13}$$

The joint angle trajectories calculated for each joint using equations (7.7)-(7.9) are plotted in Figure 7.9.

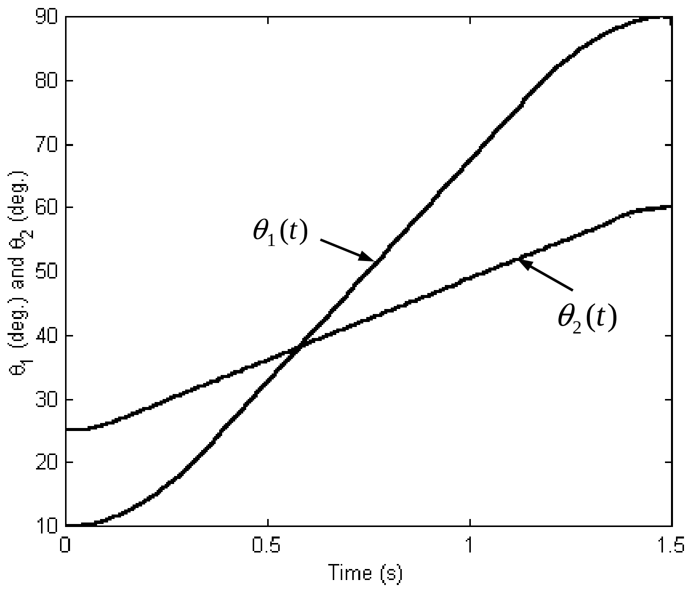

Figure 7.9 The joint angle trajectories for example 7.1.

7.4 LSPB Joint Space Trajectory Planning for Multiple Motion Segments

Typically, the programmed motion for a robot includes points in between the start point and final end point (or destination). These intermediate points are known as "via points". An example of the need to use via points to avoid collisions between the robot and obstacles in its environment is shown in Figure 7.10.

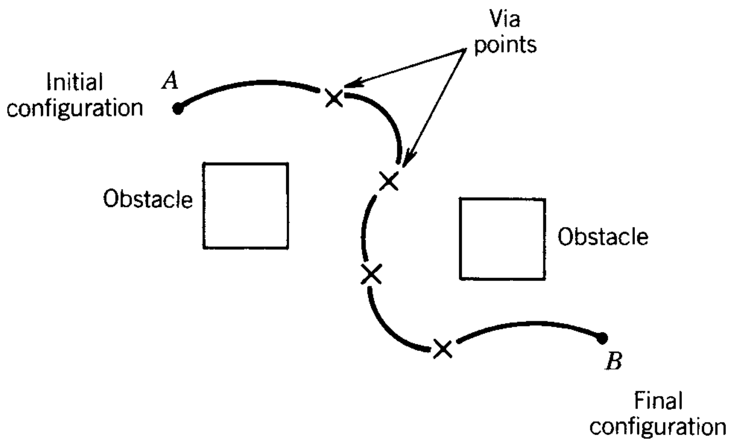

Figure 7.10 Example of using via points to avoid collisions [3].

To achieve smooth motion the trajectory will not actually pass through the via points (in fact to pass through them would require infinite acceleration). However, if the desired velocities are relatively small the error between the via points and the trajectory will also be small. An example of a multiple segment LSPB trajectory for a single joint is shown in Figure 7.11. The via points are points B, C and D. The trajectory is the thicker line shown. Craig [2] presents a method for planning such trajectories in his chapter 7. This is beyond the scope of this course.

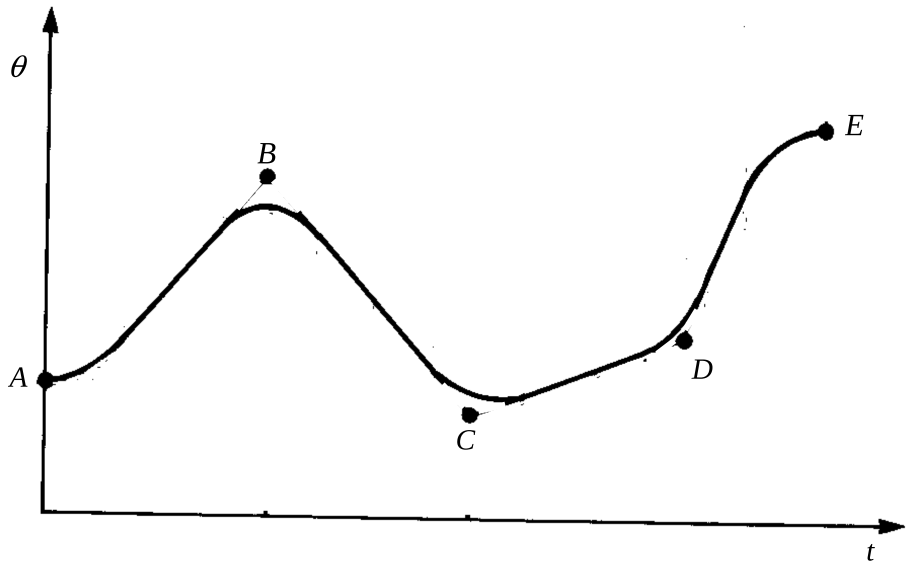

Figure 7.11 An example of a four segment trajectory planned using the LSPB method [2].

7.5 How Joint Space Trajectory Planning is Implemented

The motion of a robot occurs as a result of the following sequence of actions (assuming the user has already created a program of instructions for the robot):

1. User commands robot controller to begin executing the user program.
2. Robot controller obtains information about current and next motion segments from user program.
3. The Cartesian space descriptions of the segment points are converted to joint space using the inverse kinematics equations.
4. The trajectory planning software (part of the robot controller) generates desired position and velocity (and sometimes acceleration) values for the joints for the current time instant t.
5. The desired position and velocity values are sent to the servo control loops that control the robot's motors.
6. The time t is incremented by Δt seconds, where $\Delta t \approx 0.005$ seconds.
7. If $t = t_f$ for the current motion segment then continue with step 8, else go to step 4.
8. Go back to step 2 until the current motion segment is the last segment from the user program.

7.6 Cartesian Space Trajectory Planning

The same schemes used for joint space trajectory planning may be used for Cartesian space trajectory planning (*i.e.* third-order polynomials, fifth-order polynomials, and LSPB). The main implementation difference is the inverse kinematics calculations must be done before the desired position and velocity values may be sent to the motor control loops. This adds significantly to the calculations that must be performed every Δt seconds. It also introduces the disadvantages described in section 7.2.

Since the usual reason for choosing Cartesian space trajectory planning is to move along a straight line at a constant speed, the most common approach is LSPB. The Cartesian position information is handled the same way as the joint variables are in joint space. That is, the initial and final values of X, Y and Z, the desired accelerations, and the desired move duration (or the desired speed) are used with equations (7.6)-(7.9) to generate trajectories for each of X, Y and Z. Dealing with the Cartesian orientation information is more difficult. The rotation portion of the ${}^{0}T_{6}$ matrix is not in a form well suited to trajectory generation. So another representation of orientation must be used. Recalling section 3.3, two possible options are Euler angles and Roll, Pitch and Yaw angles. Both represent the spatial orientation of the end-effector by three angles. So the solution is to convert the orientation information for the start and end points into one of these three angle representations, and then use equations (7.6)-(7.9) to generate a trajectory for each angle.

Multiple motion segment Cartesian space LSPB trajectories will also be blended as described in section 7.4.

References

1. S.B. Niku, "Introduction to Robotics", Pearson Education, 2001.
2. J.J. Craig, "Introduction to Robotics", Addison Wesley Longman, 1989.
3. M.W. Spong and M. Vidyasagar, "Robot Dynamics and Control", John Wiley & Sons, 1989.