ECE5463: Introduction to Robotics

Lecture Note 9: Trajectory Generation

Prof. Wei Zhang

Department of Electrical and Computer Engineering
Ohio State University
Columbus, Ohio, USA

Spring 2018

Outline

- Trajectory Generation Problem
- Point-to-Point Trajectories
- Time Scaling of Straight-Line Path
- Trajectory Generation Using Via Points

Path and Time Scaling

- Roughly, a **trajectory** is a specification of the robot position (configuration) as a function of time.

- It is often useful to view a trajectory as a combination of a **path**, a purely geometric description of the sequence of configurations achieved by the robot, and a **time scaling**, which specifies the times when those configurations are reached.

- **Path:** a function $\theta : [0, 1] \to \Theta$ that maps a scalar path parameter s to a point in the robot configuration space Θ
  - $s = 0$: the start of the path
  - $s = 1$: the end of the path

- **Time scaling:** a function $s : [0, T] \to [0, 1]$ that maps each time instant to a value of the geometric path parameter.

Trajectory Generation Problem

- **Trajectory:** $\theta(s(t))$ or $\theta(t)$ for short.

- Velocity and acceleration along a trajectory $\theta(s(t))$:

$$\dot{\theta} = \frac{d\theta}{ds}\dot{s}; \qquad \ddot{\theta} = \frac{d\theta}{ds}\ddot{s} + \frac{d^2\theta}{ds^2}\dot{s}^2$$

- **Trajectory Generation:** Construct a trajectory (path + time scaling) so that the robot reaches a sequence of points in a given time

- Trajectory should be sufficiently smooth and respect limits on joint variables, velocities, accelerations, or torques

- Some of the joint limits can be "state-dependent"

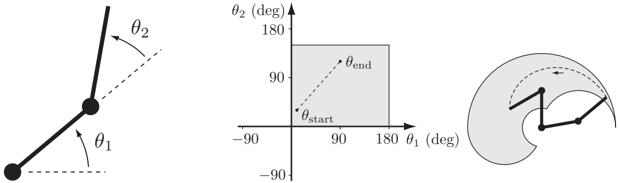
- Due to time limit, we will not formally discuss path planning problems (design path to avoid obstacles).

Straight-Line Path

- Straight-line path in joint space:

$$\theta(s) = \theta_{\text{start}} + s(\theta_{\text{end}} - \theta_{\text{start}})$$

- Straight lines in joint space generally do not yield straight-line motion of the end-effector in task space

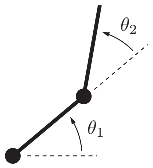


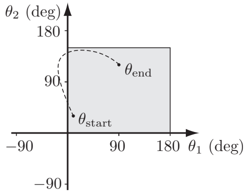
- We may prefer to have straight-line path in task space that connects $T_{\text{start}} \in SE(3)$ to $T_{\text{end}} \in SE(3)$.

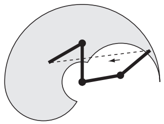
- There are two ways to design a "straight-line" path in task space: using minimum representation and using screw motion

Straight-Line Path in Task Space: Approach 1

- Let $x(s)$ be some minimum representation of the end-effect frame (e.g. location + Euler angles)
- The straight line can be defined as $x(s) = x_{\text{start}} + s(x_{\text{end}} - x_{\text{start}})$
- Use inverse kinematics to find the corresponding joint space path $\theta(s)$
- Even when x_{start} and x_{end} are both reachable, some points along the straight line in task space may not be reachable







Straight-Line Path in Task Space: Approach 2

- How do define “straight line” in $SE(3)$? Note that: $T_{\text{start}} + s(T_{\text{end}} - T_{\text{start}})$ may not lie in $SE(3)$
- Straight line motion in Euclidean space $\leftrightarrow$ Screw motion in $SE(3)$
- Let $X(s)$ be the screw motion path with constant twist such that $X(0) = T_{\text{start}}$ and $X(1) = T_{\text{end}}$.

$$X(s) = T_{\text{start}} \exp\left(\log\left(T_{\text{start}}^{-1} T_{\text{end}}\right) s\right)$$

- The translational and rotational parts of $X(s) = (R(s), p(s))$ are:

$$p(s) = p_{\text{start}} + s(p_{\text{end}} - p_{\text{start}}),\ R(s) = R_{\text{start}} \exp\left(\log\left(R_{\text{start}}^{T} R_{\text{end}}\right) s\right)$$

Time Scaling: Cubic Polynomial

- We can design scaling function $s : [0, T] \to [0, 1]$ to ensure a "smooth" motion along the path $\theta(s)$ and all the velocity and acceleration constraints are satisfied.

- We often require zero velocity at the start and the end of a path. This leads to the following constraint:

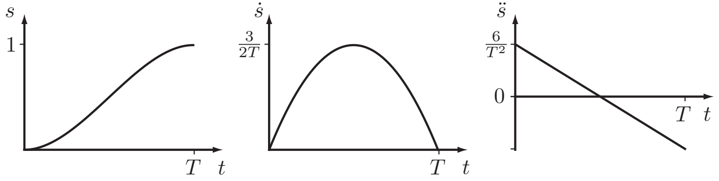
$$s(0) = 0, s(T) = 1, \dot{s}(0) = \dot{s}(T) = 0$$

- We can use a cubic polynomial to meet these constraints:

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$\Rightarrow$

Time Scaling: Cubic Polynomial



- Assume straight-line path in joint space (other cases are similar)

$$\theta(s) = \theta_{\text{start}} + s(\theta_{\text{end}} - \theta_{\text{start}})$$

- Max joint velocities are achieved at the halfway point of the motion $t = T/2$

- Max joint accelerations and decelerations are achieved at $t = 0$ and $t = T$:

- Joint constraints $\dot{\theta}_{\text{limit}}$ and $\ddot{\theta}_{\text{limit}}$ can be checked at these critical times to ensure feasibility

Time Scaling: Fifth-Order Polynomial

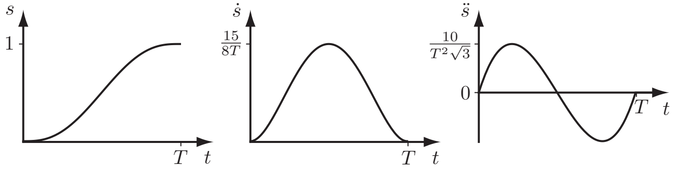
- We may also want to have zero accelerations at $t = 0$ and $t = T$, which leads to the following constraints:

$$s(0) = 0, s(T) = 1, \dot{s}(0) = \dot{s}(T) = \ddot{s}(0) = \ddot{s}(T) = 0$$

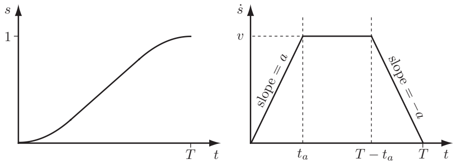
- These can be met with a fifth-order polynomial:

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

- We can use the 6 constraints to solve uniquely for the coefficients.



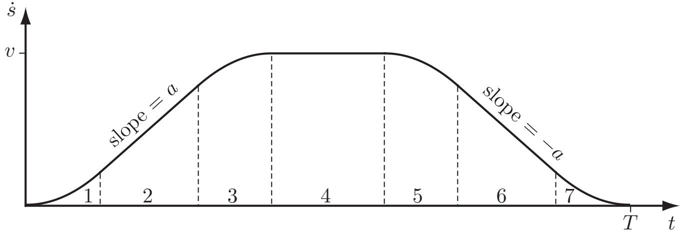
Time Scaling: Trapezoidal



- Trapezoidal time scaling:
  - $t \in [0, t_a]$: constant acceleration phase $\ddot{s} = a$
  - $t \in (t_a, T - 2t_a]$: constant velocity phase $\dot{s} = v$
  - $t \in (T - 2t_a, T]$: constant deceleration phase $\ddot{s} = -a$
- Not very smooth, but allows for easy incorporation of joint velocity and acceleration limits.
- For example, a and v can be chosen to be the largest ones satisfying

$$|(\theta_{\text{end}} - \theta_{\text{start}})v| \leq \dot{\theta}_{\text{limit}} \quad \text{and} \quad |(\theta_{\text{end}} - \theta_{\text{start}})a| \leq \ddot{\theta}_{\text{limit}}$$

Time Scaling: S-Curve



- Trapezoidal scaling causes discontinuities in acceleration at $t = 0, t_a, T - 2t_a, T$
- An improved version is the **S-curve** time scaling. It consists of 7 phases.
  - (1) constant jerk $d^3s/dt^3 = J$; (2) constant acceleration $\ddot{s} = a$;
  - (3) constant negative jerk $-J$; (4) coasting at constant v;
  - (5) constant negative jerk; (6) constant deceleration; (7) constant positive jerk

Trajectory Generation Using Via Points

- Suppose that the robot joints need to pass through a series of **via points** at specified times

- Suppose that there are no strict specifications on the shape of the path between consecutive points

- In this case, we can directly construct the trajectory $\theta(t)$ instead of finding a path $\theta(s)$ and then a time scaling $s(t)$,

- Since the trajectory for each joint can be designed individually, we focus on a single joint variable and call it $\beta(t)$ to simplify notation

Trajectory Generation Using Via Points

- Suppose there are k via points $(\beta_i,\ \dot{\beta}_i,\ T_i)$, $i = 1, \ldots, k$, where β_i and $\dot{\beta}_i$ are the specified desired position and velocity, and T_i is the specified time to reach the i^{th} via point.

- We can use a cubic polynomial to generate the ith trajectory segment.
  Polynomial: $\beta(T_i + \Delta t) = a_{i,0} + a_{i,1}\Delta t + a_{i,2}\Delta t^2 + a_{i,3}\Delta t^2$
  Constraints: $\beta(T_i) = \beta_i, \beta(T_{i+1}) = \beta_{i+1}, \dot{\beta}(T_i) = \dot{\beta}_i, \dot{\beta}(T_{i+1}) = \dot{\beta}_{i+1}$

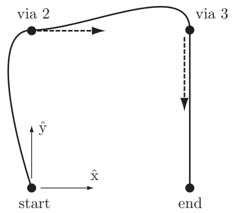
- We can solve for all the coefficients for each segment (see page 286 of textbook)

- This approach can be easily extended to 5th-order polynomial

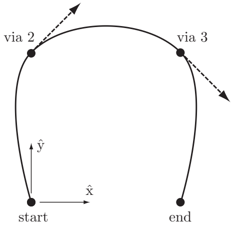
Trajectory Generation Using Via Points

Example:

- Time instants: $T_1 = 0, T_2 = 1, T_3 = 2, T_4 = 3$;
- Via points: $(0,0)$, $(0,1)$, $(1,1)$, and $(1,0)$
- Velocities: $(0,0)$, $(1,0)$, $(0,-1)$, and $(0,0)$



- Change via point velocities to $(0,0)$, $(1,1)$, $(1,-1)$, and $(0,0)$ will result in a different trajectory



More Discussions

-