31390 Unmanned Autonomous Systems - Trajectory Generation

David Wuthier

June 17, 2020

GOALS

To understand:

- How to translate a path into a trajectory
- What are the constraints behind trajectory generation
- How to generate trajectories from scratch

The path planning problem (definitions)

$$
\begin{aligned}
\mathcal{C} &: \quad \text{robot configuration space, e.g., } \mathbb{R}^2 \times \mathbb{R} \times \mathbb{S} \\
q \in \mathcal{C} &: \quad \text{robot configuration} \\
\mathcal{W} \subset \mathbb{R}^N &: \quad \text{robot workspace with } N = 2 \text{ (planar) or } N = 3 \text{ (volume)} \\
\mathcal{O} \subset \mathcal{C} &: \quad \text{obstacle region} \\
A(q) \subset \mathcal{W} &: \quad \text{robot geometry} \\
\mathcal{C}_{\text{obs}} = \{q \in \mathcal{C} \mid A(q) \cap \mathcal{O} \neq \emptyset\} &: \quad \text{C-space obstacle region} \\
\mathcal{C}_{\text{free}} = \mathcal{C} \setminus \mathcal{C}_{\text{obs}} &: \quad \text{free space}
\end{aligned}
$$

The path planning problem (formulation)

Given an initial configuration q_I and a goal configuration q_G, find

$$c : [0,1] \to \mathcal{C}_{\text{free}} \text{ such that } c(0) = q_I \text{ and } c(1) = q_G.$$

For example (linear path): $c(\lambda) = (1-\lambda)q_I + \lambda q_G$.

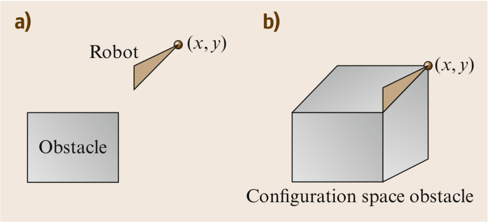

Figure 1: a) A planar robot and an obstacle. b) the corresponding obstacle region

Paths and trajectories

- A path is geometric object; it does not entail any temporal information
- How to follow a path then? With a **trajectory**
- Let's say you want to follow a path c from time t_I to time t_G. Then, you will follow the desired trajectory

$$q_d(t) = c(s(t))$$

with a scaling function $s : [t_I, t_G] \to [0, 1]$ that allows to respect kinematic and dynamic constraints.

Robot dynamics

In general, the dynamical model of a robot can be expressed as

$$\underbrace{H(q)}_{\text{inertia}}\,\ddot{q} + \underbrace{C(q,\dot{q})}_{\substack{\textit{Coriolis+}\\ \textit{centrifugal}}}\,\dot{q} + \underbrace{\tau_g(q)}_{\text{gravity}} = \underbrace{\tau}_{\text{input torque}}$$

This dynamics, together with the actuators' performances, structural aspects, safety, etc., can imply constraints like

$$|\tau| \leq \tau_{\mathsf{max}}, \quad |\dot{q}| \leq \dot{q}_{\mathsf{max}}, \quad |\ddot{q}| \leq \ddot{q}_{\mathsf{max}},$$

And the smoother the trajectory, the better for the robot.

Robot control

One way to follow trajectories is *inverse dynamics control*: set the input torque to

$$\tau = H(q)v + C(q, \dot{q})\dot{q} + \tau_g(q)$$

with an auxiliary variable v that can be set to

$$v = \ddot{q}_d + K_V(\dot{q}_d - \dot{q}) + K_P(q_d - q) \quad \text{(PD controller)}$$

Generating trajectories

If no path is available, a simple way to generate trajectories is to use **splines**. A spline is a piecewise polynomial function:

$$p(t) = \begin{cases} p_0(t) \text{ if } t_0 \leq t < t_1 \\ p_1(t) \text{ if } t_1 \leq t < t_2 \\ \vdots \\ p_m(t) \text{ if } t_m \leq t < t_{m+1} \end{cases} \qquad \text{with} \quad p_j(t) = \sum_{i=0}^{n} a_{ij} t^i$$

Trapezoidal velocity profile ($n = 2$)

When specifying, e.g., waypoints, peak velocity and acceleration

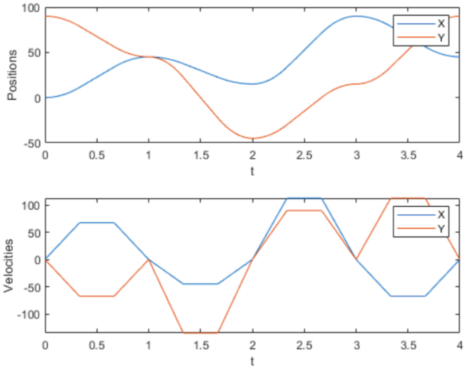

Figure 2: Trapezoidal velocity profiles.

Cubic splines ($n = 3$)

When specifying, e.g., waypoints and velocities at these waypoints.

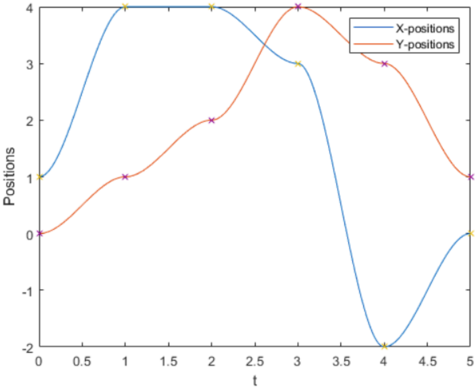

Figure 3: Trajectory constituted by cubic splines.

Quintic splines ($n = 5$)

When specifying, e.g., waypoints, velocities at these waypoints and acceleration at these waypoints.

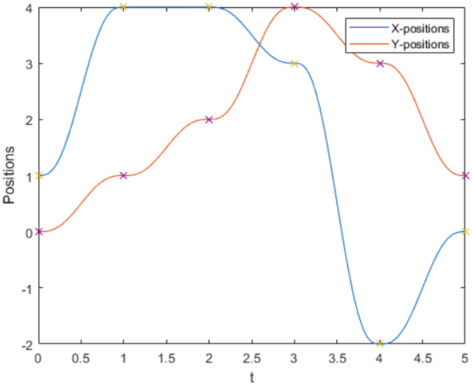

Figure 4: Trajectory constituted by quintic splines.

B-splines ($n = 1, 2, 3, \ldots$)

When specifying control points instead of waypoints.

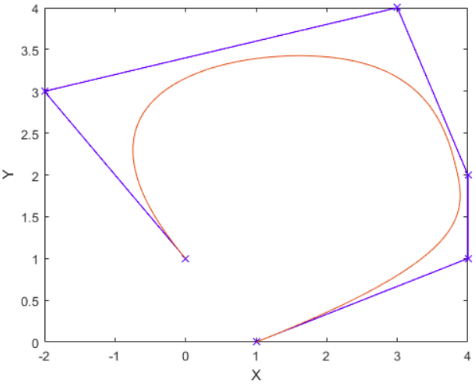

Figure 5: Trajectory constituted by a b-spline. The crosses represent control points, and the trajectory remain in the control polygon (in blue).

B-splines

A B-spline is obtained by linearly combining a set of basis functions $N_{i,p}$ using control points P_i, $i = 1, 2, \ldots, n$:

$$p(t) = \sum_{i=1}^{n} P_i N_{i,p}(t).$$

These basis can be computed with the Cox-de Boor recursion formula for $t_0 \leq t_1 \leq \ldots \leq t_m$:

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t).$$

B-splines

The computation of the basis functions follows a pyramid scheme:

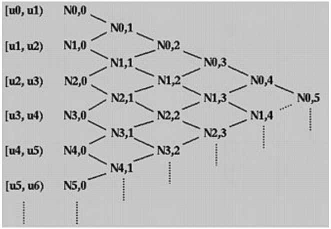

Figure 6: The de Boor algorithm.

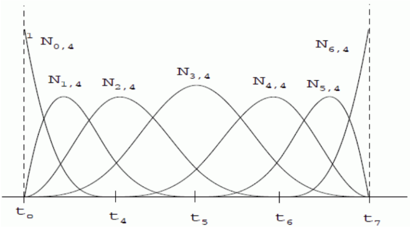

Figure 7: A set of basis functions.

Higher order?

Let's see in Mellinger and Kumar (2011)

REFERENCES

- B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.
- The robotics systems toolbox. [Online]. Available: https://se.mathworks.com/help/robotics/index.html?s_tid=CRUX_lftnav
- B-spline curve. [Online]. Available: https://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node17.html
- B-spline (wolfram). [Online]. Available: https://mathworld.wolfram.com/B-Spline.html
- B-spline basis functions. [Online]. Available: https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/bspline-basis.html
- D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2520–2525.
- T. Lee, M. Leok, and N. H. McClamroch, "Control of complex maneuvers for a quadrotor uav using geometric methods on se (3)," *arXiv preprint arXiv:1003.2005*, 2010.