MOTION GENERATION

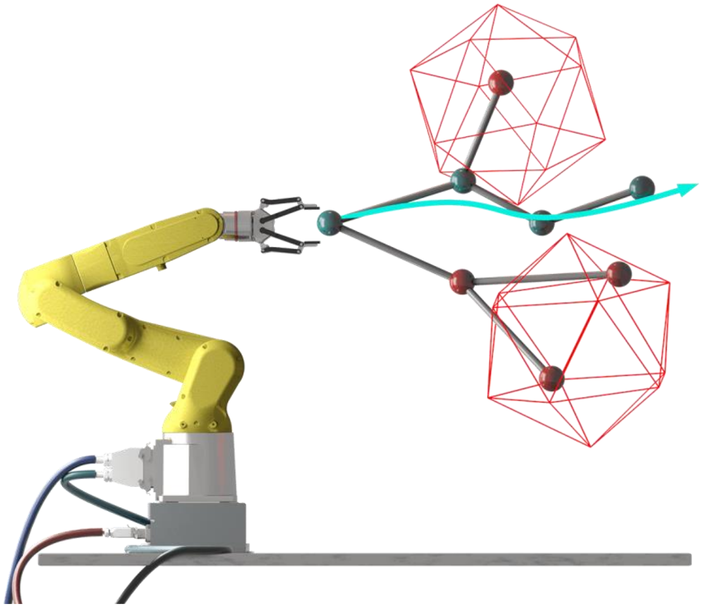
RBT350

Roberto Martin-Martin

Assistant Professor of Computer Science.

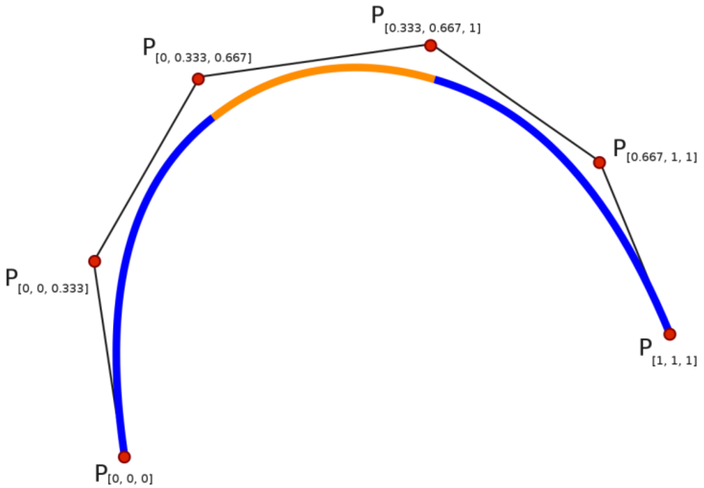
Recap

- How do we go from A to B without collisions?
  - Motion planning
- Two alternatives:
  1. We discretize the configuration space, convert it into a graph and search in the graph
  2. We sample the configuration space either covering the entire space uniformly and then creating a graph (PRM) or growing trees from the start to the goal (RRT)

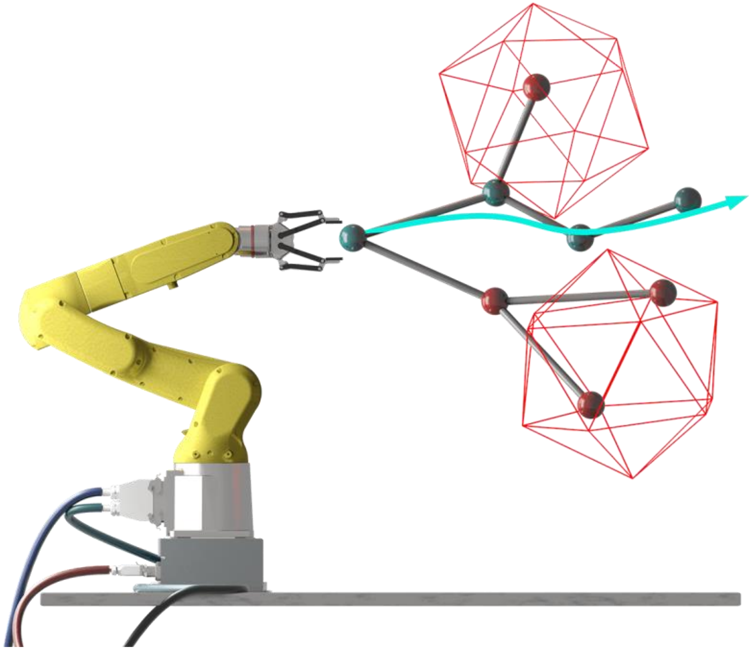


What will you learn today?

- So far, when we want to get from A to B:
  - Motion planning gives us a path
  - A path is [x1, …, xn] or [q1, …, qn], a sequence of collision-free **states/configurations** to go from A to B
- How do we execute that path with our robot?
  - What are the actions to go through those states in a smooth way? → motion generation
  - What do we do in-between? → Interpolation

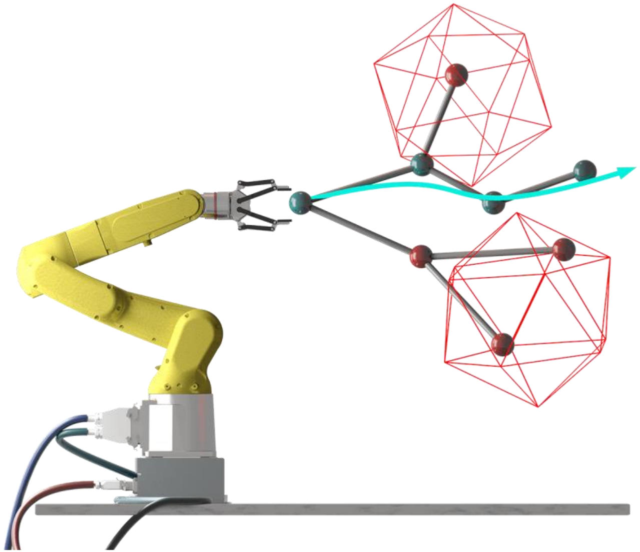


How do we execute a path?




RUN 1
Press Esc to exit full screen mode.
10
Mikaela SHIFFRIN
USA
2.4
Screencast-O-Matic.com

How do we execute a path?



- We need to find what to do in between the points

interpolation

- We want to have smooth motion

Not only positions

Motion generation!

$$[q_d[0], \ldots, q_d[N]] \rightarrow q_d(t), \dot{q}_d(t), \ddot{q}_d(t)$$

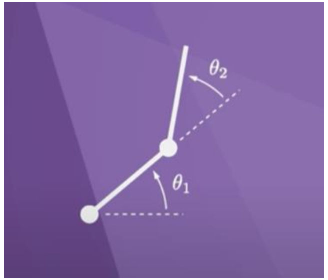
Interpolating between two configurations in joint space

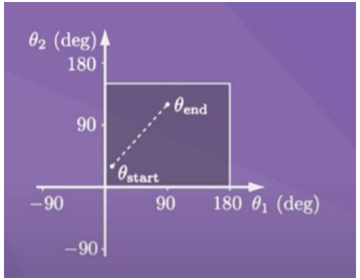
- We have an initial and a final configuration in joint space

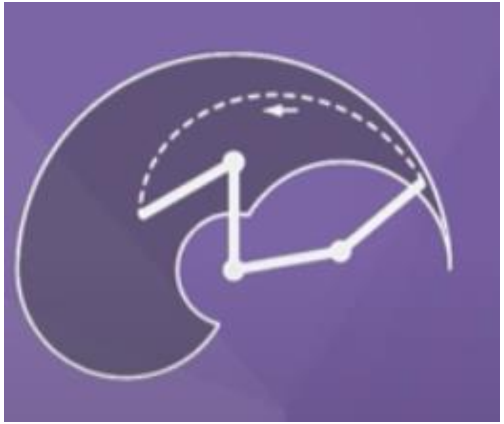
  q_{start}, q_{end}

- Linear interpolation in joint space:

  $Q(s) = q_{start} + s(q_{end} - q_{start}), s \in [0,1]$

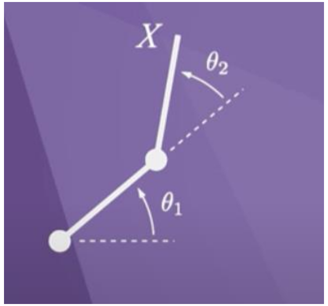


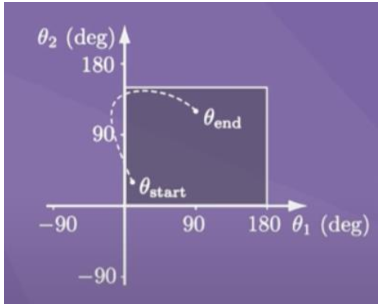




[based on Kevin Lynch slides]

Interpolating between two positions in Cartesian space





- We have an initial and a final position in Cartesian space

  x_{start}, x_{end}

Not regular +/-

- Linear interpolation in Cartesian space:

  $X(s) = x_{start} + s(x_{end} - x_{start}), s \in [0,1]$

- $IK\big(P(s)\big) = q$

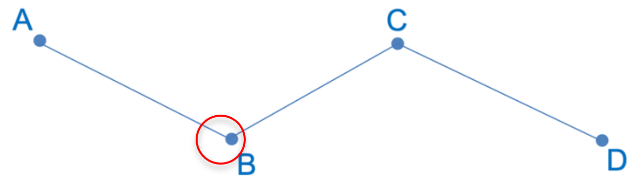
Careful! May not exist

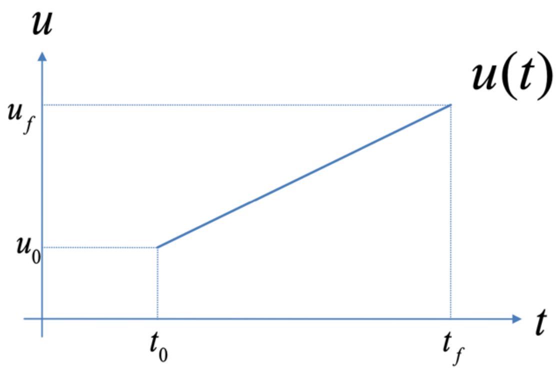
[based on Kevin Lynch slides]

Linear interpolation

- Two conditions for u
  - $u(0) = u_0$
  - $u(t_f) = u_f$

- No control over velocities!
  - Discontinuities at the beginning and end of the motion requires infinite acceleration

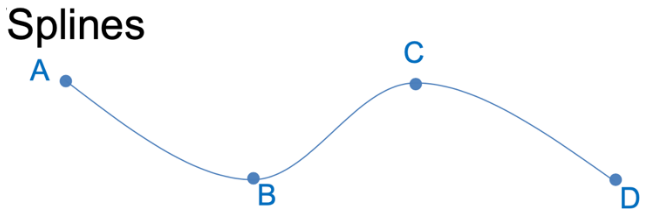




$$u(t) = a_0 + a_1 t$$

Polynomials/Spline Interpolation

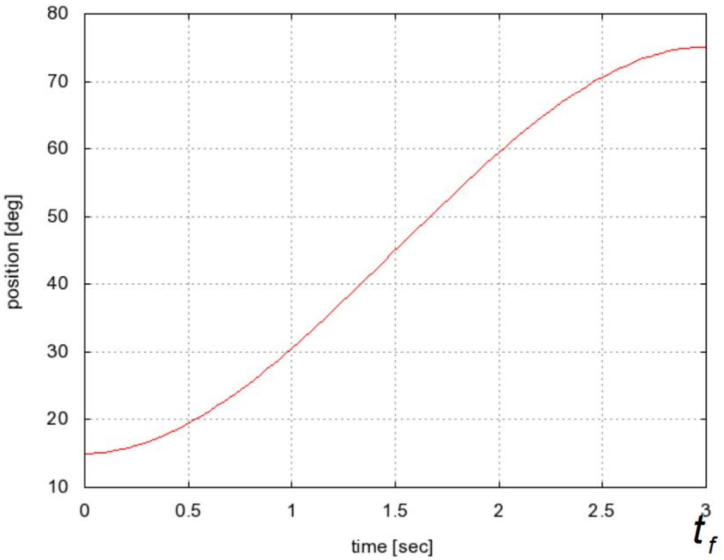
- Spline: function defined piecewise by polynomi
  - Instead of a single high degree polynomial, we h multiple lower degree polynomials for each piece
- What is the degree of each piece?
  - Depends on how many constraints we want to be able to control
  - Degrees of polynomial (n) → We can control n+1 constraints
  - Typical constraints we want to control:
    - Boundary conditions: <u>position</u>, velocity, acceleration at the via points
    - Max velocity or acceleration during the segment



Cubic Splines

- Initial pose conditions
  - $u(0) = u_0$
  - $u(t_f) = u_f$

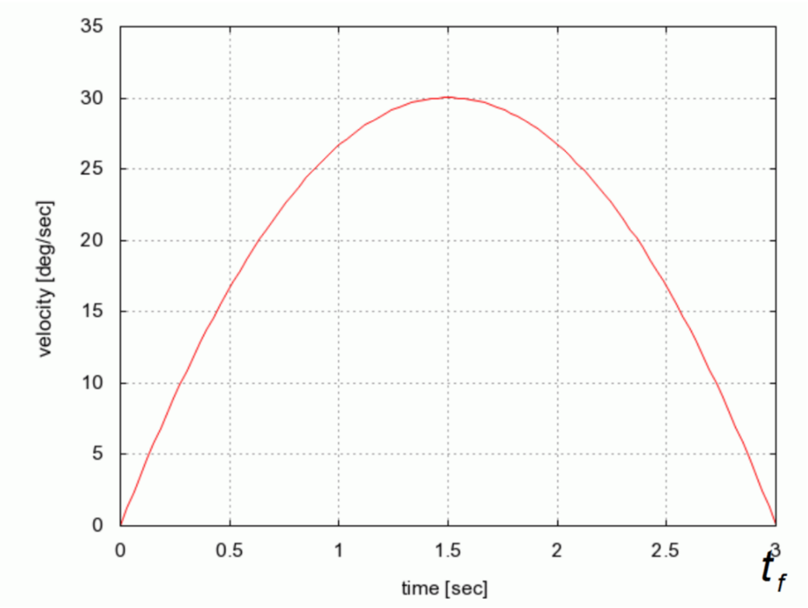
$$u(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$



Cubic Splines

- Velocity initial conditions
  - $\dot{u}(0) = 0$
  - $\dot{u}(t_f) = 0$

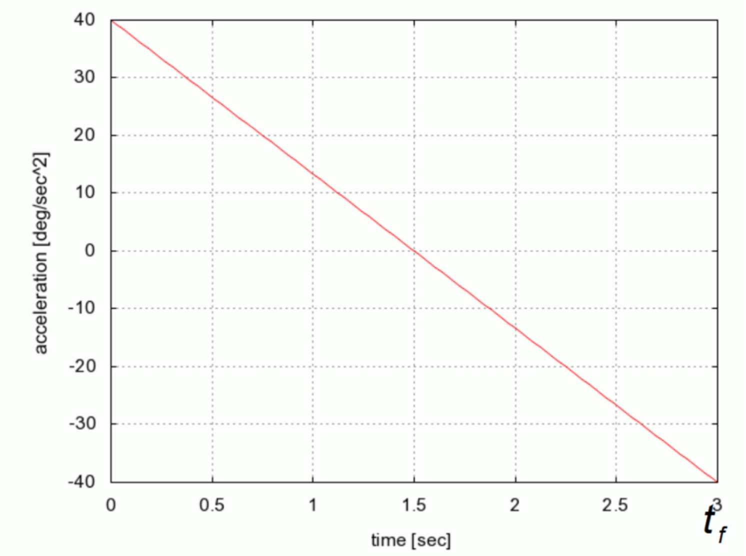
$$\dot{u}(t) = a_1 + 2a_2 t + 3a_3 t^2$$



Cubic Splines

- No control over accelerations!
- If we want to control accelerations, we need higher order polynomials (quintic, septic…)

$$\ddot{u}(t) = 2a_2 + 6a_3 t$$

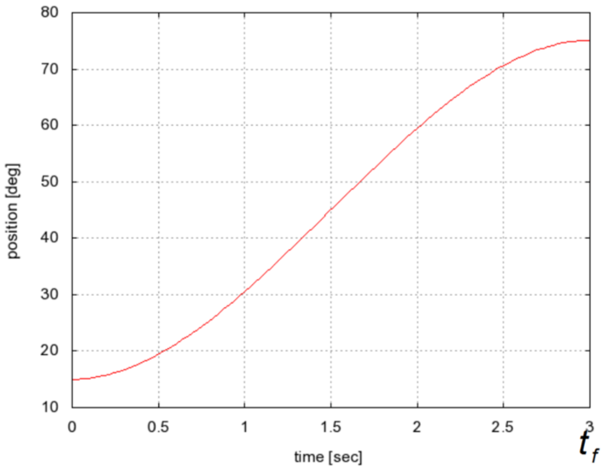


Cubic Splines

- $u(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$
- Four equations:
  - $u(0) = u_0$
  - $u(t_f) = u_f$
  - $\dot{u}(0) = 0$
  - $\dot{u}(t_f) = 0$
- Four unknowns: a_0, a_1, a_2, a_3

Cubic Splines

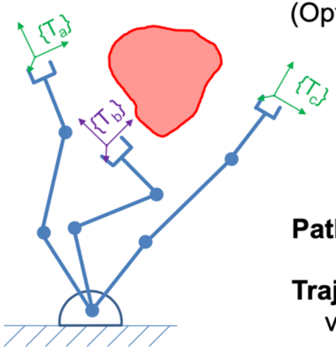
$$u(t) = u_0 + \frac{3}{t_f^2}(u_f - u_0)t^2 - \frac{2}{t_f^3}(u_f - u_0)\,t^3$$



Cubic Spline

Move the manipulator from an initial position $\{T_a\}$ to a desired final position $\{T_c\}$

(Optional: through some via point $\{T_b\}$)



Path points: initial, final and via points

Trajectory: time history of position, velocity and acceleration for each DOF

- How do we handle the via points?

Cubic Spline – Via points

- We concatenate one spline to another, providing the boundary conditions
- We assume we do not stop at the via points (smooth trajectory!)

Cubic Spline – Via points

- We concatenate one spline to another, providing the boundary conditions
- We assume we do not stop at the via points (smooth trajectory!)

$$u_1(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$u_1(0) = u_0$$
$$u_1(t_{via}) = u_{via}$$
$$\dot{u}_1(0) = \dot{u}_0$$
$$\dot{u}_1(t_{via}) = \dot{u}_{via}$$

$$u_2(t) = b_0 + b_1(t - t_{via}) + b_2(t - t_{via})^2 + b_3(t - t_{via})^3$$

$$u_2(t_{via}) = u_{via}$$
$$u_2(t_f) = u_f$$
$$\dot{u}_2(t_{via}) = \dot{u}_{via}$$
$$\dot{u}_2(t_f) = 0$$

Cubic Spline – Via points

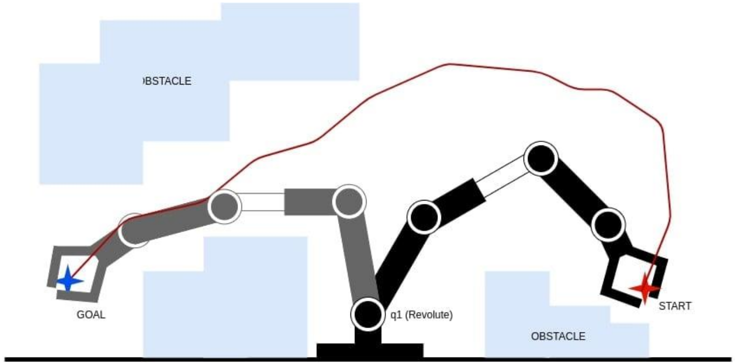
- How do we choose the velocity at the via point?
  - "User" specifies
  - Use a heuristic (e.g., half of the maximum velocity)
  - Modify the boundary conditions
    - Remove velocity constraints and impose acceleration and velocity to be continuous

How do we apply this to a robot trajectory?

- Path: $[q_0, \ldots, q_k]$ → Via points in configuration space
- Separate per joint
- Interpolate each joint independently
- How do we coordinate them?
  - Either the times are given and the same

  or

  - We compute the maximum time (limited by max vel. or acc.) and use it for all joint dimensions



What will our controller take in?

$q_d(t)$

$\dot{q}_d(t)$

$\ddot{q}(t)$

→ **Trajectory Controller** → $a(t)$

Summary

- Motion generation
  - Goes from a sparse set of points (from a motion planner) to a continuous function of goals at each time step
  - Takes into account the velocity and accelerations necessary to make the motion smooth
  - Splines: we define the trajectory over time with a polynomial and use the boundary conditions (initial and end points, velocities, …) to find the coefficients