



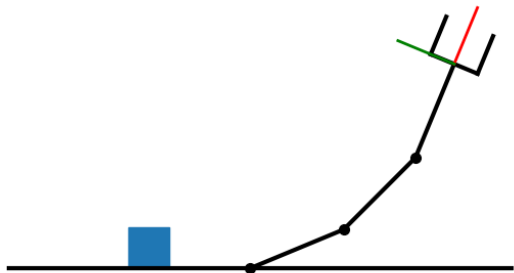
Robotics: Path and trajectory generation

Vladimír Petřík

vladimir.petrík@cvut.cz

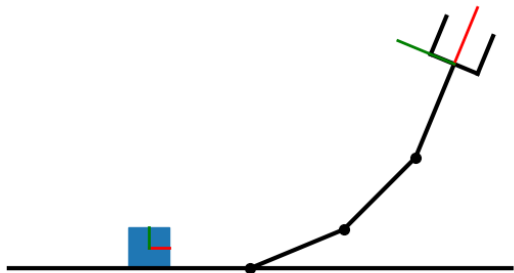
10.11.2024

Motivation: pick a cube



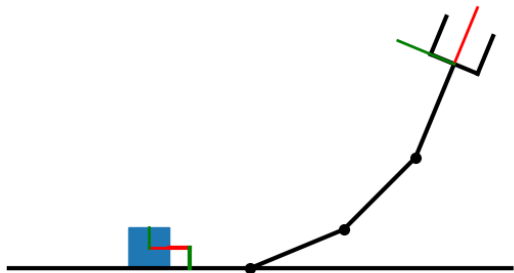
Motivation: pick a cube

- Detect where the cube is in $SE(2)$, $SE(3)$



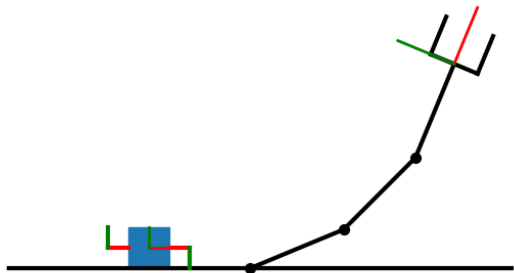
Motivation: pick a cube

- ▶ Detect where the cube is in $SE(2)$, $SE(3)$
- ▶ Define handle(s) w.r.t. cube
- ▶ Compute gripper pose



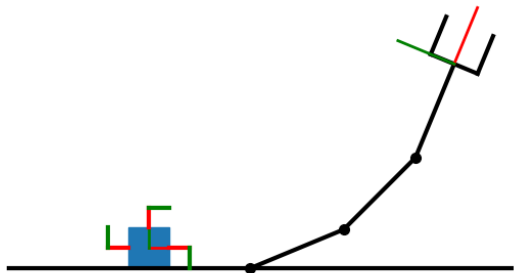
Motivation: pick a cube

- ▶ Detect where the cube is in $SE(2)$, $SE(3)$
- ▶ Define handle(s) w.r.t. cube
- ▶ Compute gripper pose



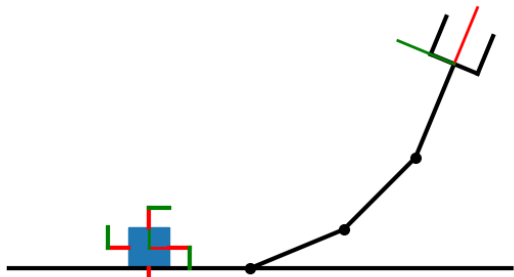
Motivation: pick a cube

- ▶ Detect where the cube is in $SE(2)$, $SE(3)$
- ▶ Define handle(s) w.r.t. cube
- ▶ Compute gripper pose



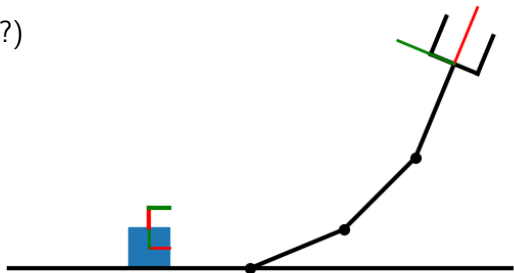
Motivation: pick a cube

- ▶ Detect where the cube is in $SE(2)$, $SE(3)$
- ▶ Define handle(s) w.r.t. cube
- ▶ Compute gripper pose



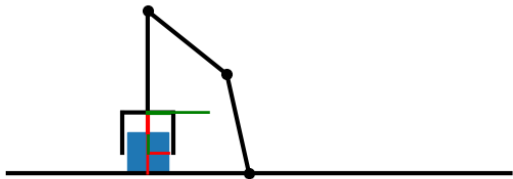
Motivation: pick a cube

- ▶ Detect where the cube is in $SE(2)$, $SE(3)$
- ▶ Define handle(s) w.r.t. cube
- ▶ Compute gripper pose
- ▶ Solve IK (select one of the solutions, how?)



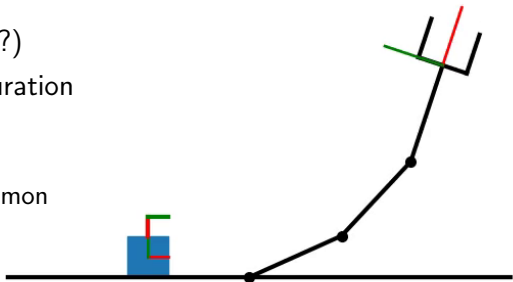
Motivation: pick a cube

- ▶ Detect where the cube is in $SE(2)$, $SE(3)$
- ▶ Define handle(s) w.r.t. cube
- ▶ Compute gripper pose
- ▶ Solve IK (select one of the solutions, how?)
- ▶ Send robot to selected joint-space configuration
- ▶ What motion will robot follow?



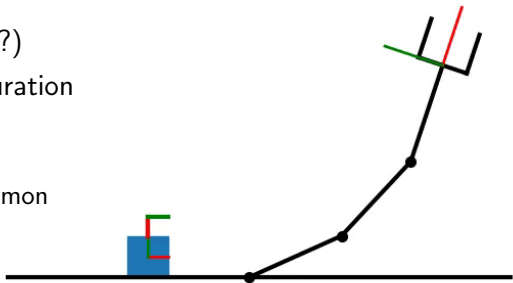
Motivation: pick a cube

- ▶ Detect where the cube is in $SE(2)$, $SE(3)$
- ▶ Define handle(s) w.r.t. cube
- ▶ Compute gripper pose
- ▶ Solve IK (select one of the solutions, how?)
- ▶ Send robot to selected joint-space configuration
- ▶ What motion will robot follow?
 - ▶ depends on the robot
 - ▶ linear interpolation in joint space is common



Motivation: pick a cube

- ▶ Detect where the cube is in $SE(2)$, $SE(3)$
- ▶ Define handle(s) w.r.t. cube
- ▶ Compute gripper pose
- ▶ Solve IK (select one of the solutions, how?)
- ▶ Send robot to selected joint-space configuration
- ▶ What motion will robot follow?
 - ▶ depends on the robot
 - ▶ linear interpolation in joint space is common
 - ▶ what is motion?



Motion

► Path

- Geometrical description (sequence of configurations)
- No timestamps, dynamics, or control restrictions
- $\mathbf{q}(s) \in \mathcal{C}_{\text{free}}, s \in [0, 1]$
- Main assumption is that trajectory can be computed by postprocessing



Motion

- ▶ Path
 - ▶ Geometrical description (sequence of configurations)
 - ▶ No timestamps, dynamics, or control restrictions
 - ▶ $\mathbf{q}(s) \in \mathcal{C}_{\text{free}}, s \in [0, 1]$
 - ▶ Main assumption is that trajectory can be computed by postprocessing
- ▶ Trajectory
 - ▶ Robot configuration in time
 - ▶ $\mathbf{q}(t) \in \mathcal{C}_{\text{free}}, t \in [0, T]$



Motion

► Path

- Geometrical description (sequence of configurations)
- No timestamps, dynamics, or control restrictions
- $\mathbf{q}(s) \in \mathcal{C}_{\text{free}}, s \in [0, 1]$
- Main assumption is that trajectory can be computed by postprocessing

► Trajectory

- Robot configuration in time
- $\mathbf{q}(t) \in \mathcal{C}_{\text{free}}, t \in [0, T]$



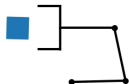
Grasping path

- ▶ Let us focus on **path** first
- ▶ Is grasping path safe? Depends on the start configuration.



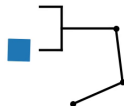
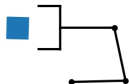
Grasping path

- ▶ Let us focus on **path** first
- ▶ Is grasping path safe? Depends on the start configuration.



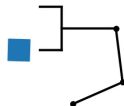
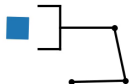
Grasping path

- ▶ Let us focus on **path** first
- ▶ Is grasping path safe? Depends on the start configuration.



Grasping path

- ▶ Let us focus on **path** first
- ▶ Is grasping path safe? Depends on the start configuration.



Pre-grasp pose

- ▶ We can define pre-grasp pose
 - ▶ e.g. 5 cm away from the object, w.r.t. handle
 - ▶ how to define 5 cm away? By design of handle.
 - ▶ fix handle orientation to have x -axis pointing towards the object
 - ▶ gripper orientation to have x -axis pointing out of gripper



Pre-grasp pose

- ▶ We can define pre-grasp pose
 - ▶ e.g. 5 cm away from the object, w.r.t. handle
 - ▶ how to define 5 cm away? By design of handle.
 - ▶ fix handle orientation to have x -axis pointing towards the object
 - ▶ gripper orientation to have x -axis pointing out of gripper
 - ▶ grasp pose T_{RH}
 - ▶ if gripper T_{RG} equals T_{RH} , object is grasped
 - ▶ pre-grasp pose $T_{RP} = T_{RH}T_x(-\delta_{\text{pre_grasp}})$



Pre-grasp pose

- ▶ We can define pre-grasp pose
 - ▶ e.g. 5 cm away from the object, w.r.t. handle
 - ▶ how to define 5 cm away? By design of handle.
 - ▶ fix handle orientation to have x -axis pointing towards the object
 - ▶ gripper orientation to have x -axis pointing out of gripper
 - ▶ grasp pose T_{RH}
 - ▶ if gripper T_{RG} equals T_{RH} , object is grasped
 - ▶ pre-grasp pose $T_{RP} = T_{RH}T_x(-\delta_{\text{pre_grasp}})$
- ▶ Is path from pre-grasp to grasp safe if $\delta_{\text{pre_grasp}}$ is small?

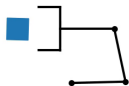


Pre-grasp pose

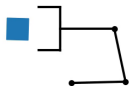
- ▶ We can define pre-grasp pose
 - ▶ e.g. 5 cm away from the object, w.r.t. handle
 - ▶ how to define 5 cm away? By design of handle.
 - ▶ fix handle orientation to have x -axis pointing towards the object
 - ▶ gripper orientation to have x -axis pointing out of gripper
 - ▶ grasp pose T_{RH}
 - ▶ if gripper T_{RG} equals T_{RH} , object is grasped
 - ▶ pre-grasp pose $T_{RP} = T_{RH}T_x(-\delta_{\text{pre_grasp}})$
- ▶ Is path from pre-grasp to grasp safe if $\delta_{\text{pre_grasp}}$ is small?
- ▶ Is path from pre-grasp to grasp safe if $\delta_{\text{pre_grasp}}$ is large?



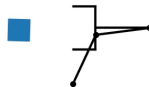
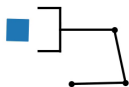
Pre-grasp pose



Pre-grasp pose



Pre-grasp pose



Interpolation in joint space

- ▶ Also called straight-line path, point-to-point path
- ▶ Start $\mathbf{q}_{\text{start}}$
- ▶ Goal $\mathbf{q}_{\text{goal}}$
- ▶ $\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}}), \quad s \in [0, 1]$
- ▶ Easy to compute, well defined

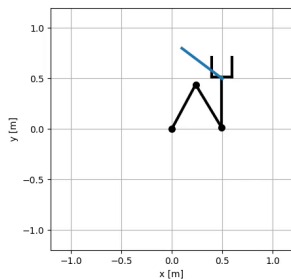


Interpolation in joint space

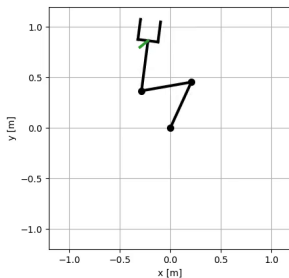
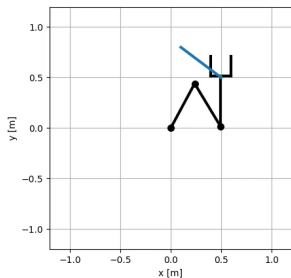
- ▶ Also called straight-line path, point-to-point path
- ▶ Start $\mathbf{q}_{\text{start}}$
- ▶ Goal $\mathbf{q}_{\text{goal}}$
- ▶ $\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}}), \quad s \in [0, 1]$
- ▶ Easy to compute, well defined
- ▶ What is the motion of the gripper?
 - ▶ likely not straight-line (for revolute joints)
 - ▶ combinations of circular paths (for revolute joints)



Interpolation in joint space



Interpolation in joint space



Interpolation in $SE(2)$ and $SE(3)$

- ▶ Straight-line path in task space
 - ▶ position $\mathbf{t}(s) = \mathbf{t}_{\text{start}} + s(\mathbf{t}_{\text{goal}} - \mathbf{t}_{\text{start}})$, $s \in [0, 1]$



Interpolation in $SE(2)$ and $SE(3)$

- ▶ Straight-line path in task space

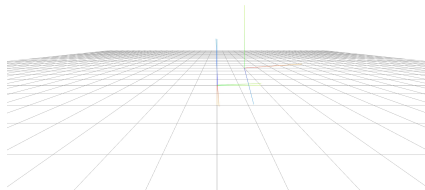
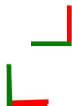
- ▶ position $\mathbf{t}(s) = \mathbf{t}_{\text{start}} + s(\mathbf{t}_{\text{goal}} - \mathbf{t}_{\text{start}})$, $s \in [0, 1]$
- ▶ rotation $R(s) = R_{\text{start}} \exp(s \log(R_{\text{start}}^{-1} R_{\text{goal}}))$, $s \in [0, 1]$



Interpolation in $SE(2)$ and $SE(3)$

- Straight-line path in task space

- position $\mathbf{t}(s) = \mathbf{t}_{\text{start}} + s(\mathbf{t}_{\text{goal}} - \mathbf{t}_{\text{start}}), \quad s \in [0, 1]$
- rotation $R(s) = R_{\text{start}} \exp(s \log(R_{\text{start}}^{-1} R_{\text{goal}})) , \quad s \in [0, 1]$



Joint-space path from task-space path

- ▶ Compute $\mathbf{q}(s)$ from $T_{RG}(s)$
- ▶ Solve IK for each s and pick the first solution of IK?



Joint-space path from task-space path

- ▶ Compute $\mathbf{q}(s)$ from $T_{RG}(s)$
- ▶ Solve IK for each s and pick the first solution of IK?
 - ▶ we did not define what is *first* solution of IK
 - ▶ let us use the closest solution of IK
 - ▶ can it happen that closest solution is not *close enough*?



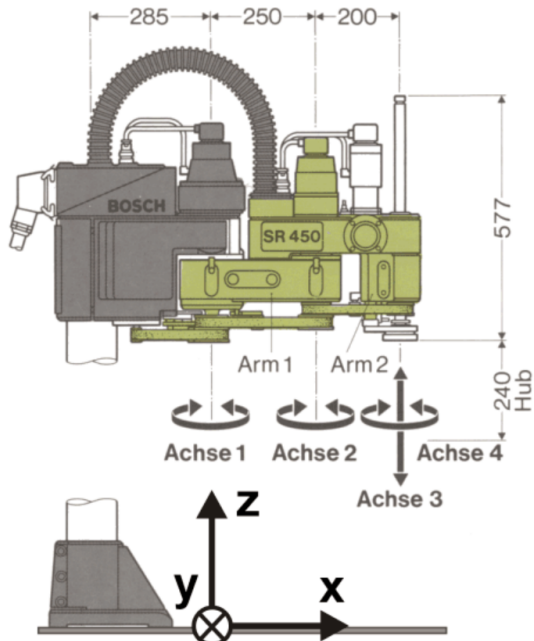
Joint-space path from task-space path

- ▶ Compute $\mathbf{q}(s)$ from $T_{RG}(s)$
- ▶ Solve IK for each s and pick the first solution of IK?
 - ▶ we did not define what is *first* solution of IK
 - ▶ let us use the closest solution of IK
 - ▶ can it happen that closest solution is not *close enough*? **yes**, let us see an example

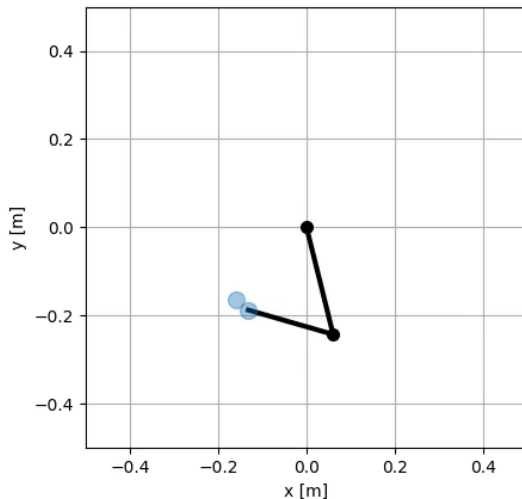


SCARA robot

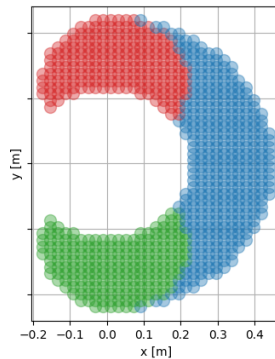
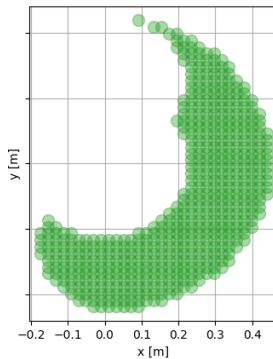
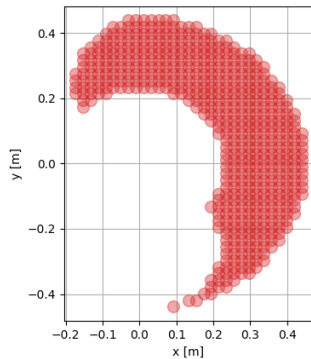
- ▶
- ▶ Analyze kinematics of SCARA
- ▶ Structure RRPR
- ▶ Self-collisions avoided by joint limits
 - ▶ $\pm 85^\circ$
 - ▶ $\pm 120^\circ$
 - ▶ $(-330 \text{ mm}, 5 \text{ mm})$
 - ▶ $(-20^\circ, 1080^\circ)$
- ▶ Compute FK and IK in xy -plane



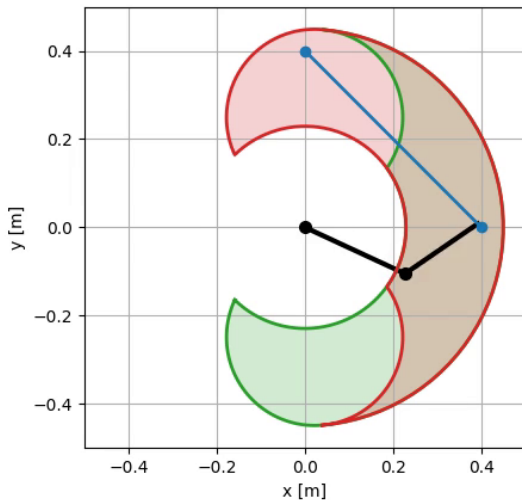
SCARA robot workspace



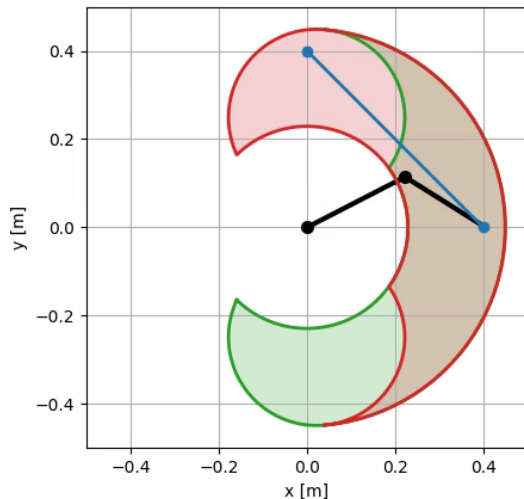
SCARA robot IK



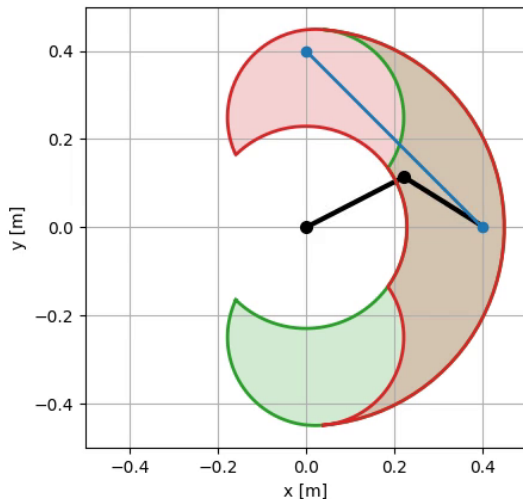
Task-space interpolation



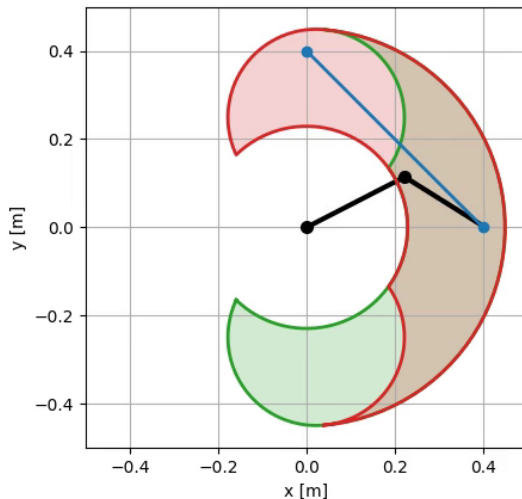
Task-space interpolation



Task-space interpolation



Task-space interpolation

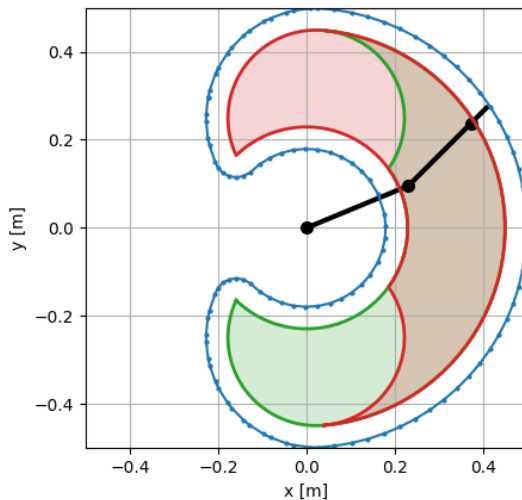


Task space interpolation

- ▶ Not all solutions of IK are available everywhere
- ▶ We need to resolve jumps in configuration space
- ▶ To change the configuration we need to pass via singularity
- ▶ The task-space interpolation can be used for pre-grasp to grasp path



SCARA effect of the last link



Trajectory from path

- ▶ Time scaling $s(t)$, $t \in [0, T]$, $s : [0, T] \rightarrow [0, 1]$
- ▶ A path and time scaling defines trajectory $\mathbf{q}(s(t))$
- ▶ Derivations:
 - ▶ velocity: $\dot{\mathbf{q}} = \frac{d\mathbf{q}}{ds} \dot{s}$
 - ▶ acceleration: $\ddot{\mathbf{q}} = \frac{d\mathbf{q}}{ds} \ddot{s} + \frac{d^2\mathbf{q}}{ds^2} \dot{s}$



Straight-line path time scaling

► Path

- position: $\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}}), \quad s \in [0, 1]$
- velocity: $\dot{\mathbf{q}} = \dot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$



Straight-line path time scaling

► Path

- position: $\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}}), \quad s \in [0, 1]$
- velocity: $\dot{\mathbf{q}} = \dot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- acceleration: $\ddot{\mathbf{q}} = \ddot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$



Straight-line path time scaling

► Path

- position: $\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}}), \quad s \in [0, 1]$
- velocity: $\dot{\mathbf{q}} = \dot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- acceleration: $\ddot{\mathbf{q}} = \ddot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$



Straight-line path time scaling

- ▶ Path

- ▶ position: $\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}}), \quad s \in [0, 1]$

- ▶ velocity: $\dot{\mathbf{q}} = \dot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$

- ▶ acceleration: $\ddot{\mathbf{q}} = \ddot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$

- ▶ 3rd order polynomial time scaling

- ▶ $s(t) = a_0 + a_1t + a_2t^2 + a_3t^3$



Straight-line path time scaling

- ▶ Path

- ▶ position: $\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$, $s \in [0, 1]$

- ▶ velocity: $\dot{\mathbf{q}} = \dot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$

- ▶ acceleration: $\ddot{\mathbf{q}} = \ddot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$

- ▶ 3rd order polynomial time scaling

- ▶ $s(t) = a_0 + a_1t + a_2t^2 + a_3t^3$

- ▶ $\dot{s}(t) = a_1 + 2a_2t + 3a_3t^2$



Straight-line path time scaling

► Path

- position: $\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$, $s \in [0, 1]$
- velocity: $\dot{\mathbf{q}} = \dot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- acceleration: $\ddot{\mathbf{q}} = \ddot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$

► 3rd order polynomial time scaling

- $s(t) = a_0 + a_1t + a_2t^2 + a_3t^3$
- $\dot{s}(t) = a_1 + 2a_2t + 3a_3t^2$
- constraints: $s(0) = \dot{s}(0) = 0$, $s(T) = 1$, $\dot{s}(T) = 0$



Straight-line path time scaling

► Path

- position: $\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$, $s \in [0, 1]$
- velocity: $\dot{\mathbf{q}} = \dot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- acceleration: $\ddot{\mathbf{q}} = \ddot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$

► 3rd order polynomial time scaling

- $s(t) = a_0 + a_1t + a_2t^2 + a_3t^3$
- $\dot{s}(t) = a_1 + 2a_2t + 3a_3t^2$
- constraints: $s(0) = \dot{s}(0) = 0$, $s(T) = 1$, $\dot{s}(T) = 0$
- solution that satisfies constraints: $a_0 = 0$, $a_1 = 0$, $a_2 = 3/T^2$, $a_3 = -2/T^3$



Straight-line path time scaling

► Path

- position: $\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$, $s \in [0, 1]$
- velocity: $\dot{\mathbf{q}} = \dot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- acceleration: $\ddot{\mathbf{q}} = \ddot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$

► 3rd order polynomial time scaling

- $s(t) = a_0 + a_1t + a_2t^2 + a_3t^3$
- $\dot{s}(t) = a_1 + 2a_2t + 3a_3t^2$
- constraints: $s(0) = \dot{s}(0) = 0$, $s(T) = 1$, $\dot{s}(T) = 0$
- solution that satisfies constraints: $a_0 = 0$, $a_1 = 0$, $a_2 = 3/T^2$, $a_3 = -2/T^3$

► Trajectory

- $\mathbf{q}(t) = \mathbf{q}_{\text{start}} + \left(\frac{3t^2}{T^2} - \frac{2t^3}{T^3} \right) (\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- $\dot{\mathbf{q}}(t) = \left(\frac{6t}{T^2} - \frac{2t^2}{T^3} \right) (\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$



Straight-line path time scaling

► Path

- position: $\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$, $s \in [0, 1]$
- velocity: $\dot{\mathbf{q}} = \dot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- acceleration: $\ddot{\mathbf{q}} = \ddot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$

► 3rd order polynomial time scaling

- $s(t) = a_0 + a_1t + a_2t^2 + a_3t^3$
- $\dot{s}(t) = a_1 + 2a_2t + 3a_3t^2$
- constraints: $s(0) = \dot{s}(0) = 0$, $s(T) = 1$, $\dot{s}(T) = 0$
- solution that satisfies constraints: $a_0 = 0$, $a_1 = 0$, $a_2 = 3/T^2$, $a_3 = -2/T^3$

► Trajectory

- $\mathbf{q}(t) = \mathbf{q}_{\text{start}} + \left(\frac{3t^2}{T^2} - \frac{2t^3}{T^3} \right) (\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- $\dot{\mathbf{q}}(t) = \left(\frac{6t}{T^2} - \frac{2t^2}{T^3} \right) (\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- $\ddot{\mathbf{q}}(t) = \left(\frac{6}{T^2} - \frac{12t}{T^3} \right) (\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$



Straight-line path time scaling

► Path

- position: $\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$, $s \in [0, 1]$
- velocity: $\dot{\mathbf{q}} = \dot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- acceleration: $\ddot{\mathbf{q}} = \ddot{s}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$

► 3rd order polynomial time scaling

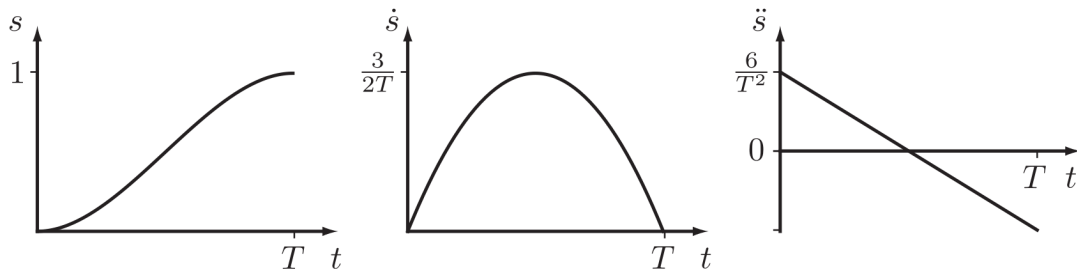
- $s(t) = a_0 + a_1t + a_2t^2 + a_3t^3$
- $\dot{s}(t) = a_1 + 2a_2t + 3a_3t^2$
- constraints: $s(0) = \dot{s}(0) = 0$, $s(T) = 1$, $\dot{s}(T) = 0$
- solution that satisfies constraints: $a_0 = 0$, $a_1 = 0$, $a_2 = 3/T^2$, $a_3 = -2/T^3$

► Trajectory

- $\mathbf{q}(t) = \mathbf{q}_{\text{start}} + \left(\frac{3t^2}{T^2} - \frac{2t^3}{T^3} \right) (\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- $\dot{\mathbf{q}}(t) = \left(\frac{6t}{T^2} - \frac{2t^2}{T^3} \right) (\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- $\ddot{\mathbf{q}}(t) = \left(\frac{6}{T^2} - \frac{12t}{T^3} \right) (\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$



3rd order polynomial time scaling



Straight-line path time scaling

- ▶ Maximum joint velocities:
 - ▶ $t = T/2$
 - ▶ $\dot{\mathbf{q}}_{\max} = \frac{3}{2T}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- ▶ Maximum joint acceleration:
 - ▶ $t = 0$ and $t = T$
 - ▶ $\ddot{\mathbf{q}}_{\max} = \left\| \frac{6}{T^2}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}}) \right\|$
 - ▶ $\ddot{\mathbf{q}}_{\min} = - \left\| \frac{6}{T^2}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}}) \right\|$
- ▶ How to use this information?



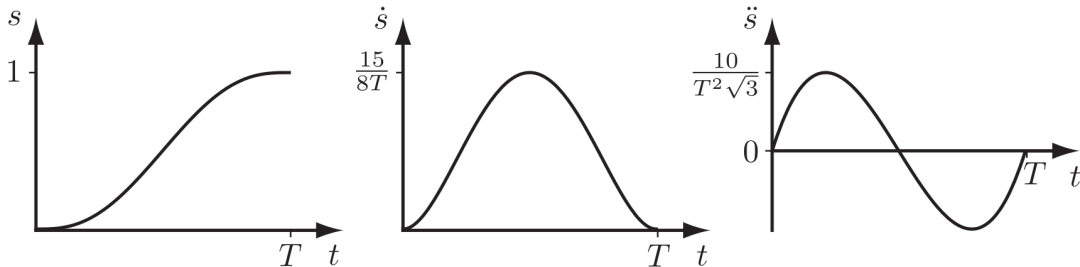
Straight-line path time scaling

- ▶ Maximum joint velocities:
 - ▶ $t = T/2$
 - ▶ $\dot{\mathbf{q}}_{\max} = \frac{3}{2T}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}})$
- ▶ Maximum joint acceleration:
 - ▶ $t = 0$ and $t = T$
 - ▶ $\ddot{\mathbf{q}}_{\max} = \left\| \frac{6}{T^2}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}}) \right\|$
 - ▶ $\ddot{\mathbf{q}}_{\min} = - \left\| \frac{6}{T^2}(\mathbf{q}_{\text{goal}} - \mathbf{q}_{\text{start}}) \right\|$
- ▶ How to use this information?
 - ▶ check if requested motion T is feasible given the velocity/acceleration limits
 - ▶ find minimum T such that velocity and acceleration constraints are satisfied



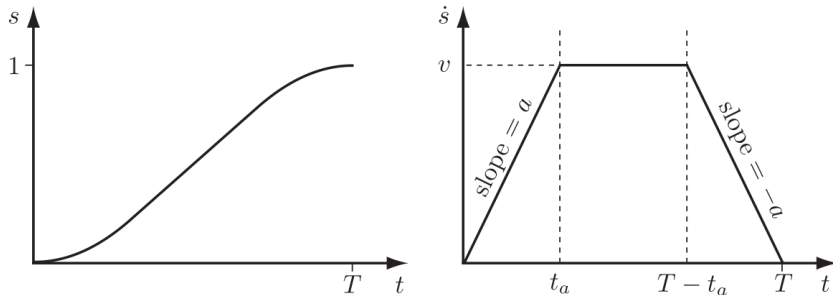
5th order polynomial

- ▶ 3rd order polynomial does not enforce zero acceleration at the beginning and end
 - ▶ infinite jerk (derivative of acceleration)
 - ▶ can cause vibrations
- ▶ We can use 5th order polynomial



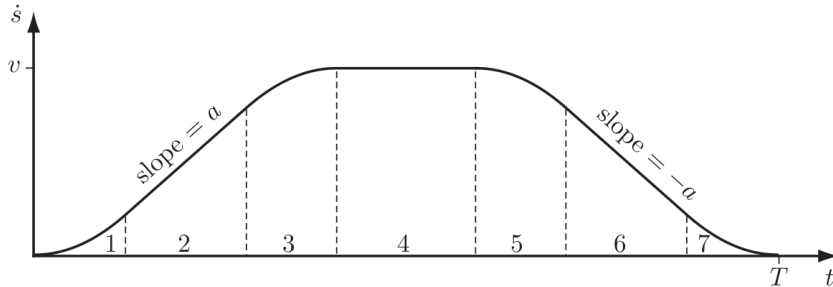
Trapezoidal time scaling

- ▶ Constant acceleration phase
- ▶ Constant velocity phase
- ▶ Constant deceleration phase
- ▶ Not smooth but it is the fastest straight-line motion possible



S-Curve time scaling

- ▶ Trapezoidal motions cause discontinuous jumps in acceleration
- ▶ S-curve smooths it to avoid vibrations
 - ▶ constant jerk, constant acceleration, constant jerk, constant velocity, constant jerk, constant deceleration, constant jerk



TAMP: Task and Motion Planning

Motivation

- ▶ We know how to plan motion for a robot in robot's configuration space
 - ▶ manually define handle on object
 - ▶ computer grasp and pre-grasp for detected object's pose
 - ▶ plan motion to pre-grasp
 - ▶ interpolate to grasp, grasp
 - ▶ interpolate to pre-grasp
 - ▶ plan motion to pre-place, place, release, pre-place



Motivation

- ▶ We know how to plan motion for a robot in robot's configuration space
 - ▶ manually define handle on object
 - ▶ computer grasp and pre-grasp for detected object's pose
 - ▶ plan motion to pre-grasp
 - ▶ interpolate to grasp, grasp
 - ▶ interpolate to pre-grasp
 - ▶ plan motion to pre-place, place, release, pre-place
- ▶ What if we have many handles? Many objects?
- ▶ **Manipulation Task and Motion Planning (TAMP)**



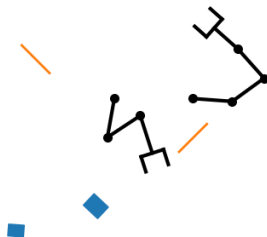
Motivation

- ▶ We know how to plan motion for a robot in robot's configuration space
 - ▶ manually define handle on object
 - ▶ computer grasp and pre-grasp for detected object's pose
 - ▶ plan motion to pre-grasp
 - ▶ interpolate to grasp, grasp
 - ▶ interpolate to pre-grasp
 - ▶ plan motion to pre-place, place, release, pre-place
- ▶ What if we have many handles? Many objects?
- ▶ **Manipulation Task and Motion Planning (TAMP)**
 - ▶ simultaneously plan task and motion solutions
 - ▶ task is the sequence of grasps and placements (discrete space)
 - ▶ motion is the sequence of robot configurations (continuous space)
 - ▶ Humanoid Path Planning (HPP) software approach



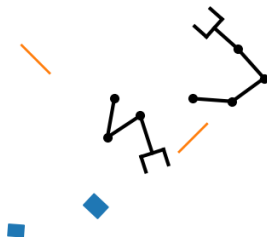
Configuration Space

- ▶ Multiple grippers connected to robots
- ▶ Environment surfaces that can be used for placing an object
- ▶ Multiple objects
 - ▶ multiple handles per object
 - ▶ multiple contact surfaces per object



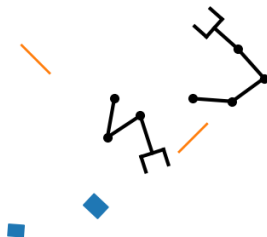
Configuration Space

- ▶ Multiple grippers connected to robots
- ▶ Environment surfaces that can be used for placing an object
- ▶ Multiple objects
 - ▶ multiple handles per object
 - ▶ multiple contact surfaces per object
- ▶ Configuration space is the set of all possible configurations of all objects and robots
 - ▶ $\mathcal{C} = \mathbb{R}^{N_1} \times \mathbb{R}^{N_2} \dots \times SE(3)^M$
 - ▶ N_i DoF of the i -th robot
 - ▶ M number of objects



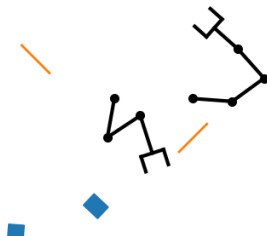
Configuration Space

- ▶ Multiple grippers connected to robots
- ▶ Environment surfaces that can be used for placing an object
- ▶ Multiple objects
 - ▶ multiple handles per object
 - ▶ multiple contact surfaces per object
- ▶ Configuration space is the set of all possible configurations of all objects and robots
 - ▶ $\mathcal{C} = \mathbb{R}^{N_1} \times \mathbb{R}^{N_2} \dots \times SE(3)^M$
 - ▶ N_i DoF of the i -th robot
 - ▶ M number of objects
 - ▶ however, not all configuration are feasible
 - ▶ constraints are used to define feasible configurations



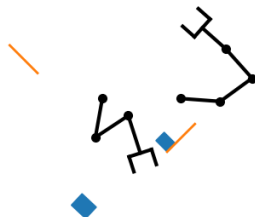
Constraints

- ▶ Object is placed or grasped, i.e. cannot fly



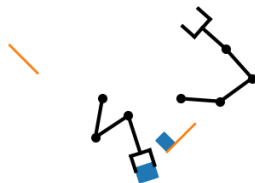
Constraints

- ▶ Object is placed or grasped, i.e. cannot fly
- ▶ Placement constraint
 - ▶ object lies on a surface
 - ▶ numerical constraints
 - ▶ **object surface** is placed on an **environment surface**



Constraints

- ▶ Object is placed or grasped, i.e. cannot fly
- ▶ Placement constraint
 - ▶ object lies on a surface
 - ▶ numerical constraints
 - ▶ **object surface** is placed on an **environment surface**
- ▶ Grasp constraint
 - ▶ object is grasped by a gripper
 - ▶ numerical constraint
 - ▶ **handle frame** equals **gripper frame**



Stav (state)

- ▶ State is a set of constraints
- ▶ Manifold of feasible configurations in the configuration space
- ▶ For example, one state can be defined by constraining both objects
 - ▶ object O_1 is placed on the surface E_1 via object surface S_1
 - ▶ object O_2 is grasped by the gripper G_1 via handle H_1

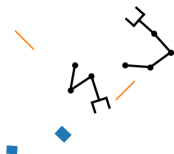


Stav (state)

- ▶ State is a set of constraints
- ▶ Manifold of feasible configurations in the configuration space
- ▶ For example, one state can be defined by constraining both objects
 - ▶ object O_1 is placed on the surface E_1 via object surface S_1
 - ▶ object O_2 is grasped by the gripper G_1 via handle H_1
- ▶ How to sample configuration from a state?
 - ▶ sample from the $\mathcal{C}$
 - ▶ geometric projection to satisfy all the constraints
 - ▶ numerical optimization (Newton-Raphson) to satisfy all the constraints



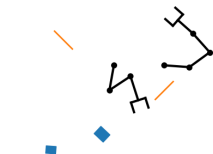
Sampling from states



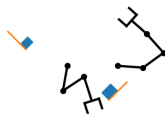
Sample from $\mathcal{C}$



Sampling from states



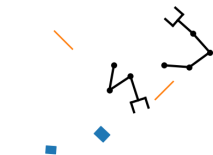
Sample from $\mathcal{C}$



- ▶ Project to state:
 - ▶ O_1 placed on E_1 via S_1
 - ▶ O_2 placed on E_2 via S_1



Sampling from states

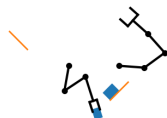


Sample from $\mathcal{C}$



► Project to state:

- O_1 placed on E_1 via S_1
- O_2 placed on E_2 via S_1



► Project to state:

- O_1 grasped by G_1 via H_1
- O_2 placed on E_2 via S_1

Transitions

- ▶ Transition defines motion between two states
 - ▶ identity transition allows to move robot inside the state
 - ▶ place transition allows to move object from the gripper to the surface
 - ▶ grasp transition allows to move object from the surface to the gripper



Transitions

- ▶ Transition defines motion between two states
 - ▶ identity transition allows to move robot inside the state
 - ▶ place transition allows to move object from the gripper to the surface
 - ▶ grasp transition allows to move object from the surface to the gripper
- ▶ Sampling on transitions vs sampling on states
 - ▶ transition respect constraints from the given state
 - ▶ for example, identity on place state will not move object (sampling on state can move object)
 - ▶ grasp transition is specified to move via pre-grasp
 - ▶ place transition is specified to move via pre-place



Interpolation on transition

- Interpolate between two configurations but respect constraints of the states/transition



Identity on place

Interpolation on transition

- Interpolate between two configurations but respect constraints of the states/transition



Identity on place



Identity on grasp

Interpolation on transition

- Interpolate between two configurations but respect constraints of the states/transition



Grasp

Interpolation on transition

- Interpolate between two configurations but respect constraints of the states/transition



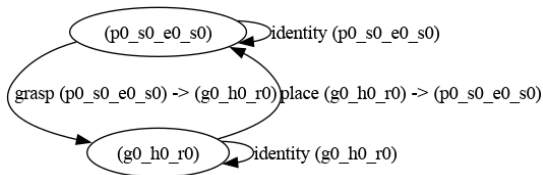
Grasp



Place

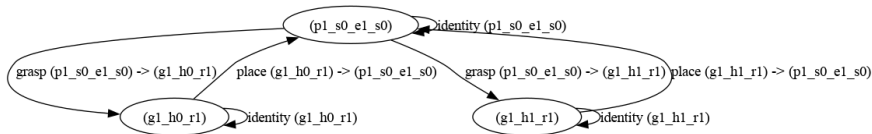
Constraint graph

- ▶ Defines all possible transitions between existing states
- ▶ Example: single arm, one object



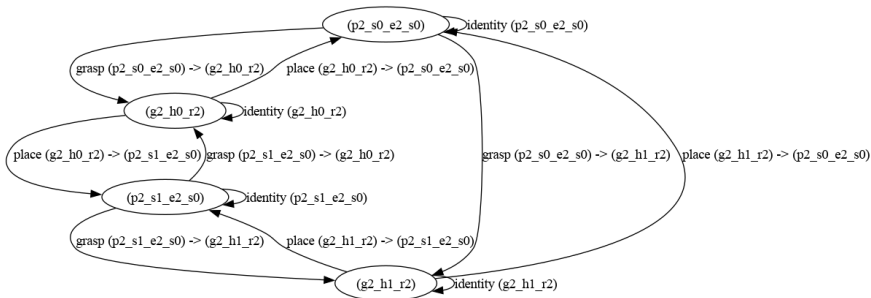
Constraint graph

- ▶ Defines all possible transitions between existing states
- ▶ Example: single arm, one object



Constraint graph

- ▶ Defines all possible transitions between existing states
- ▶ Example: single arm, one object

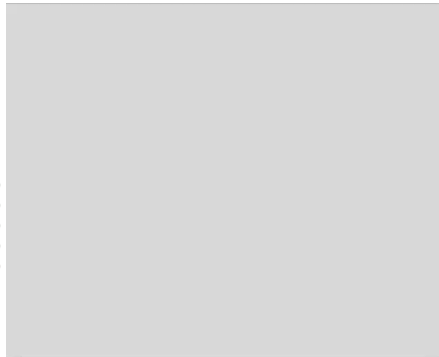
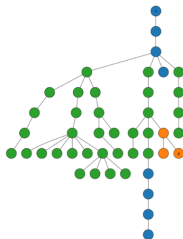
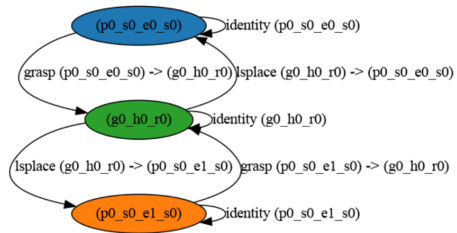


RRT on constraint graph

- ▶ Random sampling q_{rand}
 - ▶ sample random transition
 - ▶ select random existing configuration from the transition source
 - ▶ sample random configuration from the transition target reachable from beginning
- ▶ Nearest neighbor q_{tree}
 - ▶ node that is closest to q_{rand} via interpolation on the transition
- ▶ Local planner uses interpolation on transition



RRT on constraint graph



Conclusion

- ▶ Configuration space for TAMP is complex
 - ▶ discrete set of states
 - ▶ continuous motion
 - ▶ encoded by constraint graph that allow us to use RRT



Conclusion

- ▶ Configuration space for TAMP is complex
 - ▶ discrete set of states
 - ▶ continuous motion
 - ▶ encoded by constraint graph that allow us to use RRT
- ▶ Usually not used in industry
 - ▶ task space sequence is hard-coded by programmers
 - ▶ only motion is found by motion planners (if cannot be hard-coded)

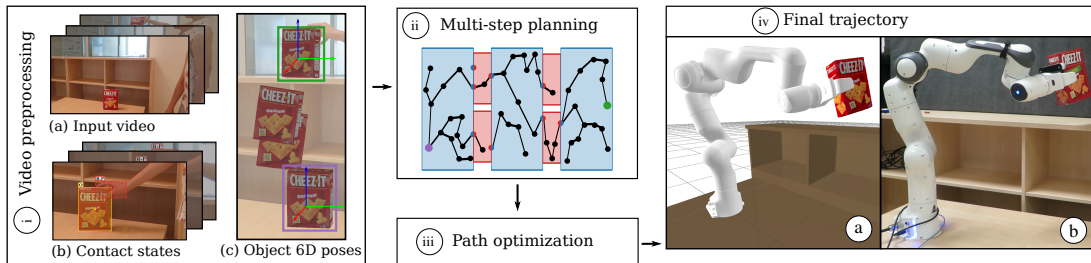


Conclusion

- ▶ Configuration space for TAMP is complex
 - ▶ discrete set of states
 - ▶ continuous motion
 - ▶ encoded by constraint graph that allow us to use RRT
- ▶ Usually not used in industry
 - ▶ task space sequence is hard-coded by programmers
 - ▶ only motion is found by motion planners (if cannot be hard-coded)
- ▶ How to avoid hard-coding? Video demonstration.

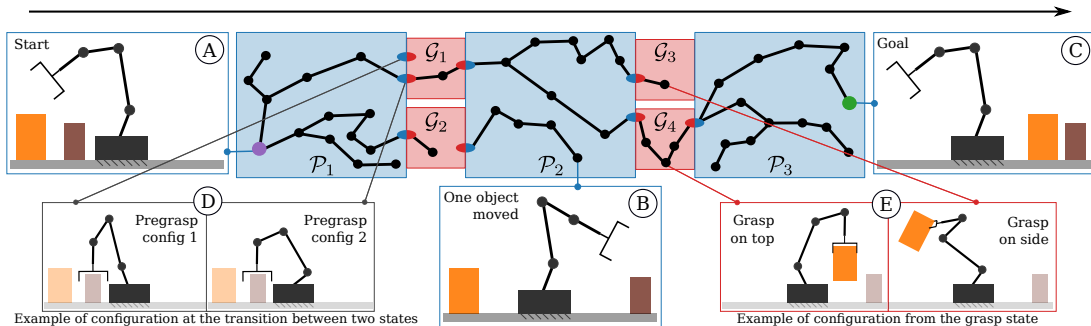


TAMP guided by video



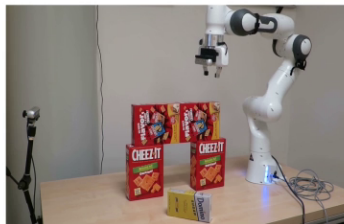
TAMP guided by video

Time



Multi-Contact Task and Motion Planning Guided by Video Demonstration

Kateryna Zorina ♣ David Kovar ♣ Florent Lamiraux ◇ Nicolas Mansard ◇
Justin Carpentier ♥ Josef Sivic ♣ Vladimir Petrik ♣



♣ CIIRC, Czech Technical University in Prague
◇ LAAS-CNRS, Université de Toulouse, CNRS, Toulouse
♥ INRIA, Paris

Summary

- ▶ Path/Trajectory
- ▶ Grasping path generation
- ▶ Interpolation in joint space and task space
- ▶ Time scaling parameterization
- ▶ TAMP



Laboratory (KN)

- ▶ Poslední oraganizované cvičení
- ▶ Implementace PRM
- ▶ Domácí úkol: implementace RRT a Random shortcut [dobrovolné]

