UC San Diego

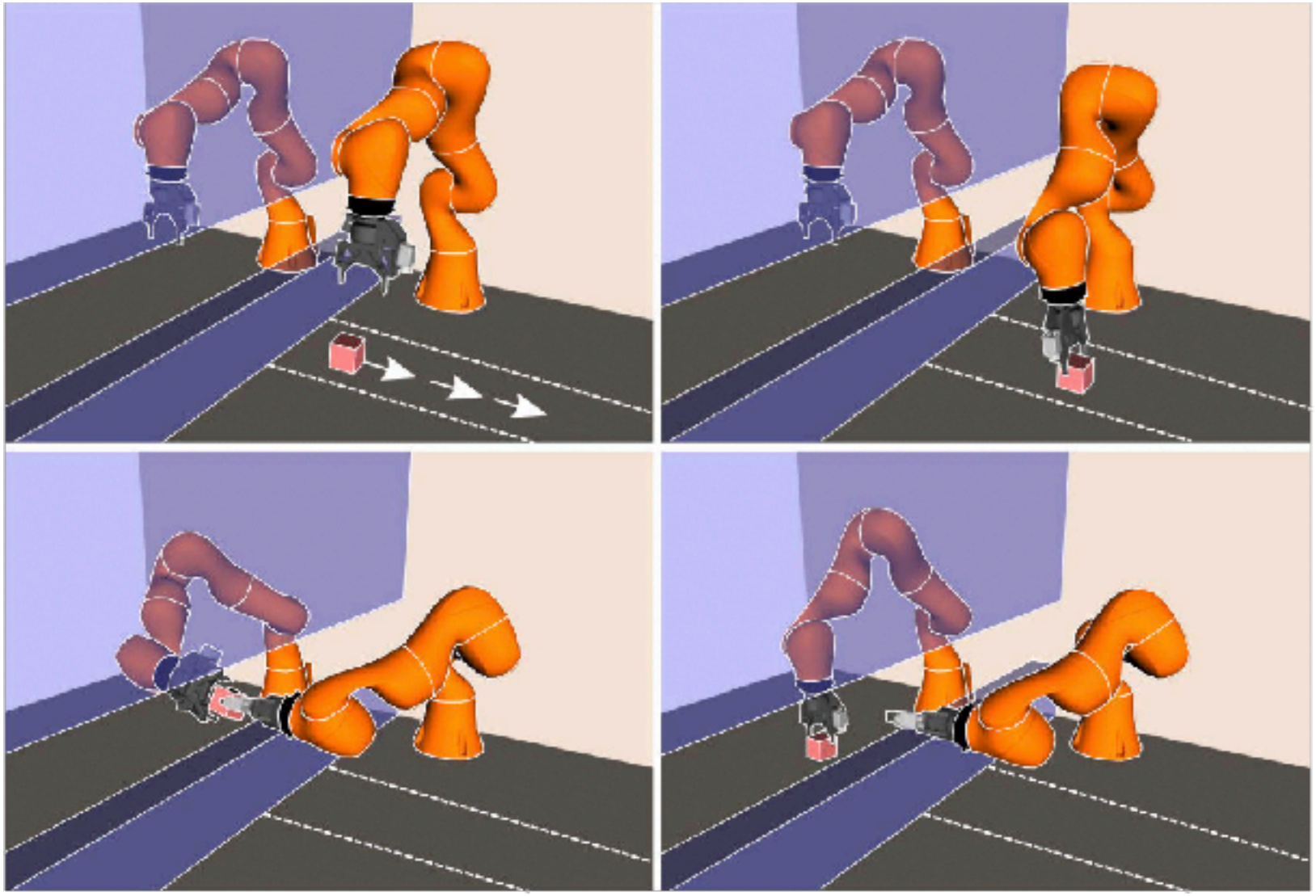
Modeling and Planning Manipulation in Dynamic Environments

Presenter: Yiming Zhao
2020.5.12

Outline

- Introduction
- Related work
- Method
- Experiments
- Conclusion

Example



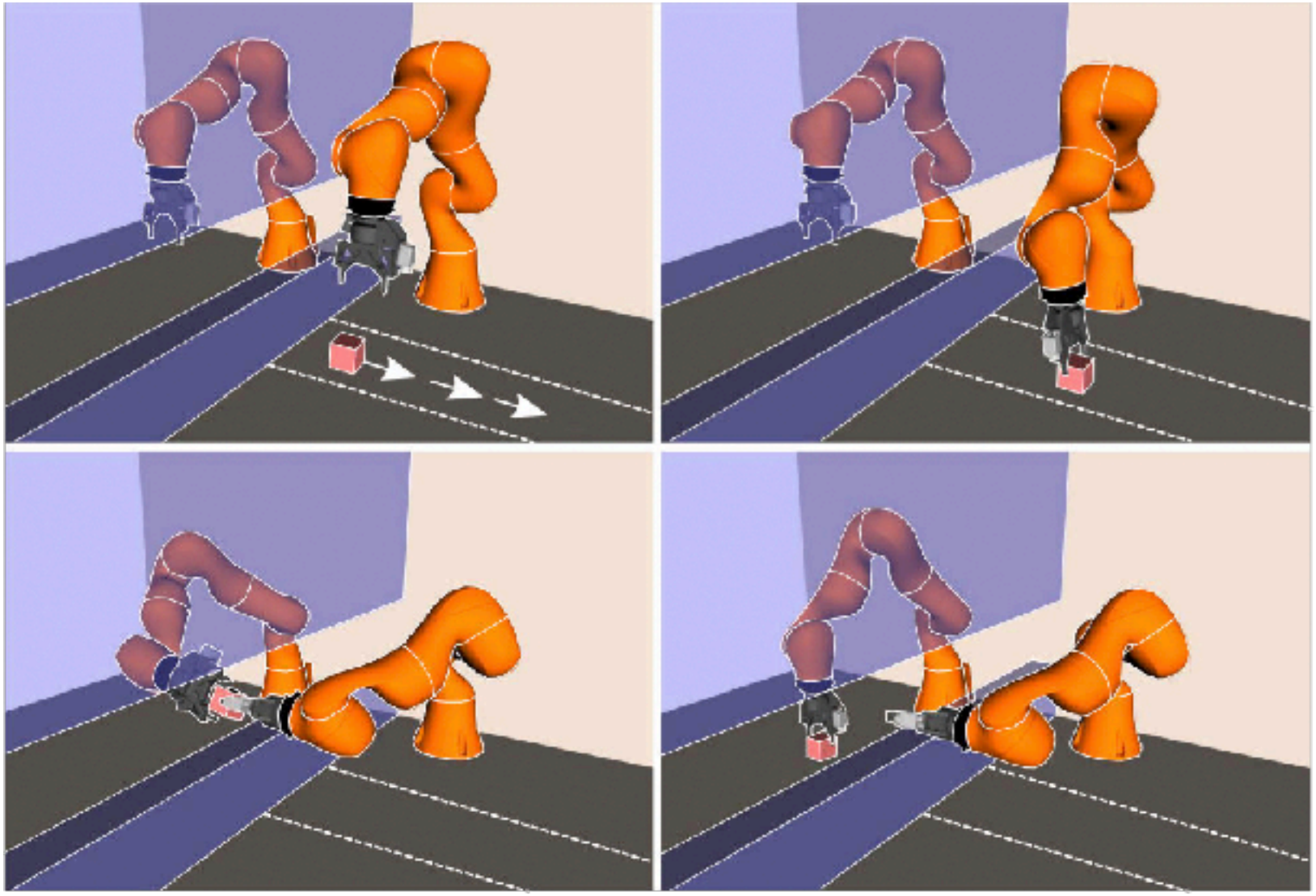
Introduction

- Property of Tasks
  - Variety of tasks
    - Example: moving conveyer belt, collaboration between robots
  - Dynamic and uncertain environments
    - Noisy environment
    - Need instantaneous reactions
  - Sequential interdependence of motions and actions
    - Subtle geometric and kinematics difference requires entirely different motion sequence

Related Works

- Constrained model based on or related to:
  - J. De Schutter, T. De Laet, J. Rutgeerts, W. Decre´, R. Smits, E. Aertbelie¨n, K. Claes, and H. Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *The Int. Journal of Robotics Research*, vol. 26, no. 5, pp. 433–455, 2007.
  - R. Alami, T. Simeon, and J.-P. Laumond, "A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps," in *The fifth Int. Symposium on Robotics Research*. MIT Press, 1990, pp. 453–463.
  - J. Mirabel and F. Lamiraux, "Manipulation planning: addressing the crossed foliation issue," in *Int. Conf. on Robotics and Automation*. IEEE, 2017, pp. 4032–4037.
- Controller based on or related to:
  - E. Aertbelie¨n and J. De Schutter, "eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs," in *Int. Conf. on Intelligent Robots and Systems*. IEEE, 2014, pp. 1540–1546.
- Search Algorithm
  - D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kino- dynamic motion planning with moving obstacles," *The Int. Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

Example

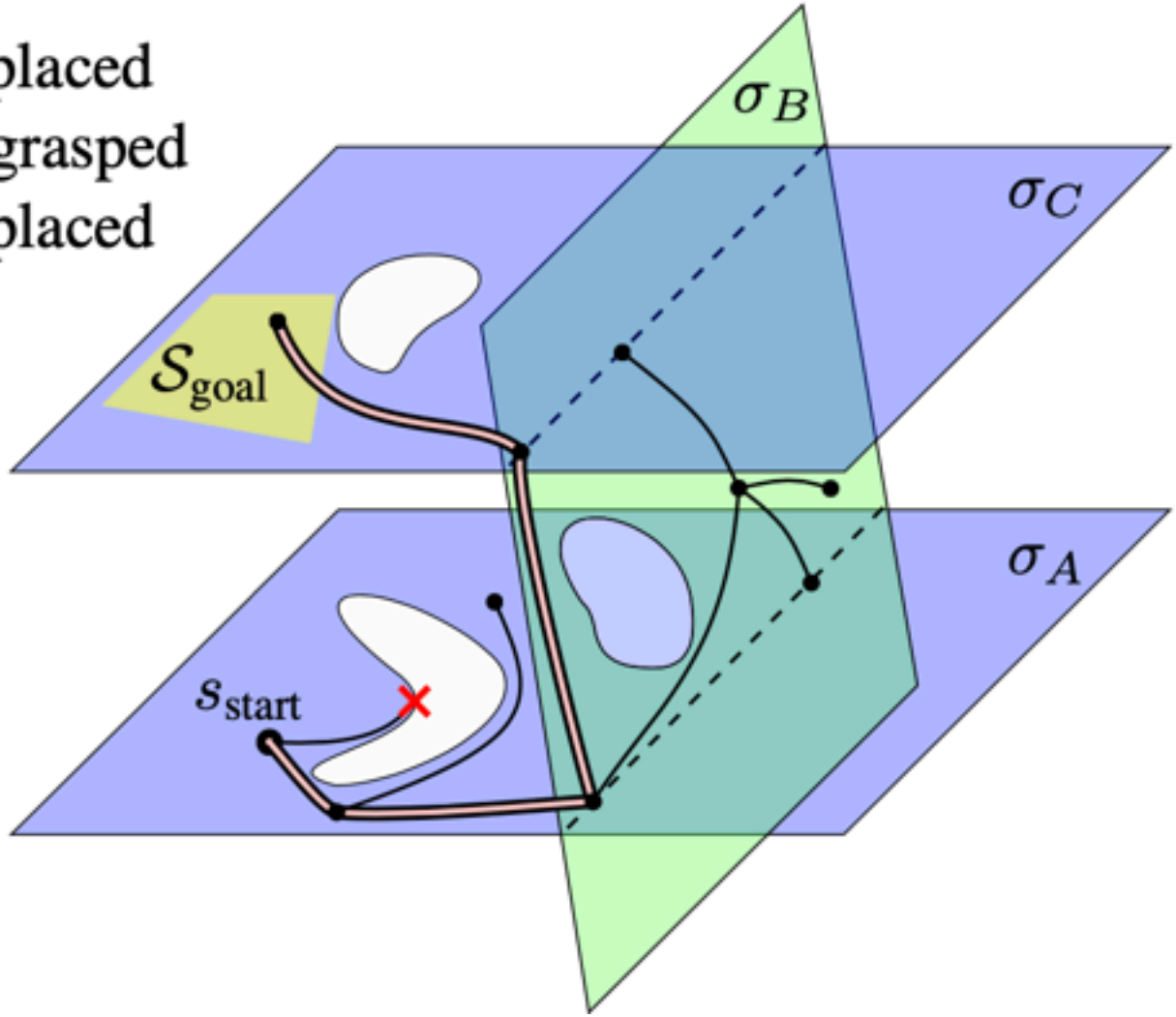


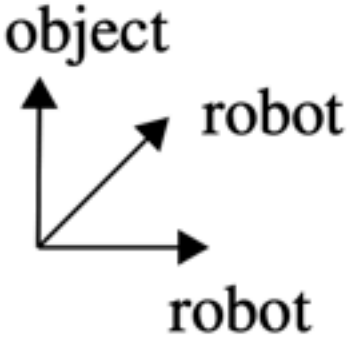
System State

- Dynamic constrained graph
  - A configuration $q \in \mathbb{R}^n$
    - Including DOF of robot grippers and the cube
    - 7 DOF for each robot, 6 DOF for the cube, 20 DOF in total
  - A discrete mode $\sigma \in \Sigma$
    - Indicator about which face of the object is in contact with which surface (surface, conveyer, hand(s))
    - There are 24 modes in total in the example
    - There are different constrains and dynamics for different modes
    - If we can change from one mode to another one, then there is an edge between these two modes, thus making it a graph
  - Full State of the System: $s = (\sigma, q, \dot{q}, t)$

Example

σ_A: object placed
σ_B: object grasped
σ_C: object placed





Constraints on States

- $f_\sigma(q,t) = 0$ e.g., when a cube is on a surface, its height should be a fixed height since it cannot go up or down

- $g_\sigma(q,t) \leq 0$ e.g.,collision-free constraint: we use pairwise penetration depths to denote distance between objects. Ideally the depth should be negative so that no object collides with other objects

- $v_\sigma(q,\dot{q},t) = 0$: time-derivative constraint: e.g., when a robot is grasping the cube, at contact points, the relative velocity between the cube and the robot gripper should be 0

- $\omega_\sigma(q,\dot{q},t) \leq 0$: time-derivative constraint: e.g., Axis-velocity-limits are given as inequality constraints.

Goal

- Goal of the planner: to find a finite sequence of controllers $\{\pi_i\}_{1 \leq i \leq k}$ that steer the system from a start state s_{start} to a goal region s_{goal}

Affine Dynamics Model

- Vector u as input (14 robot axis accelerations)
- $\ddot{q} = a_\sigma(q, \dot{q}, t) + B_\sigma(q, \dot{q}, t)u, \qquad u_{min} \leq u \leq u_{max}$

- Note:
  - q includes both object (cube) and gripper configuration
  - But u only includes gripper axis acceleration

- Why affine?:
  - In case of a grasp, the acceleration of its pose is determined by the forward-kinematics of the grasping robot.
  - When the cube is placed on a surface its acceleration is zero.
  - If an object is placed on a conveyor belt, the equation is time dependent.

UC San Diego

Kinodynamic Planner

- randomController(σ): returns a **controller** that **steers towards** a randomly chosen robot configuration satisfying the constraints of mode σ

- π = randomController(σ) can be simulated by simulateController(s,π), which will return the state s_{new} to which the system is moved by the random controller

- Similarly, transitionController(σ, σ_{new}) returns a controller π that steers towards the intersection of both mode's constraint manifolds while satisfying the constraints of σ

Procedure: buildSearchTree(s_{start}) *infinite version*

$N \leftarrow \{s_{\text{start}}\}, \quad E \leftarrow \{\}$

while *true* **do**

  $s \leftarrow (\sigma, q, \dot{q}, t) \leftarrow \text{sampleWeighted}(N)$

  $\pi \leftarrow \text{randomController}(\sigma)$

  $s_{\text{new}} \leftarrow \text{simulateController}(s, \pi)$

  $N.\text{add}(s_{\text{new}}), \quad E.\text{add}((s, \pi, s_{\text{new}}))$

Explore on the same mode

  for $\sigma' \in \mathcal{N}(\sigma)$ **do**

    $\pi \leftarrow \text{transitionController}(\sigma, \sigma')$

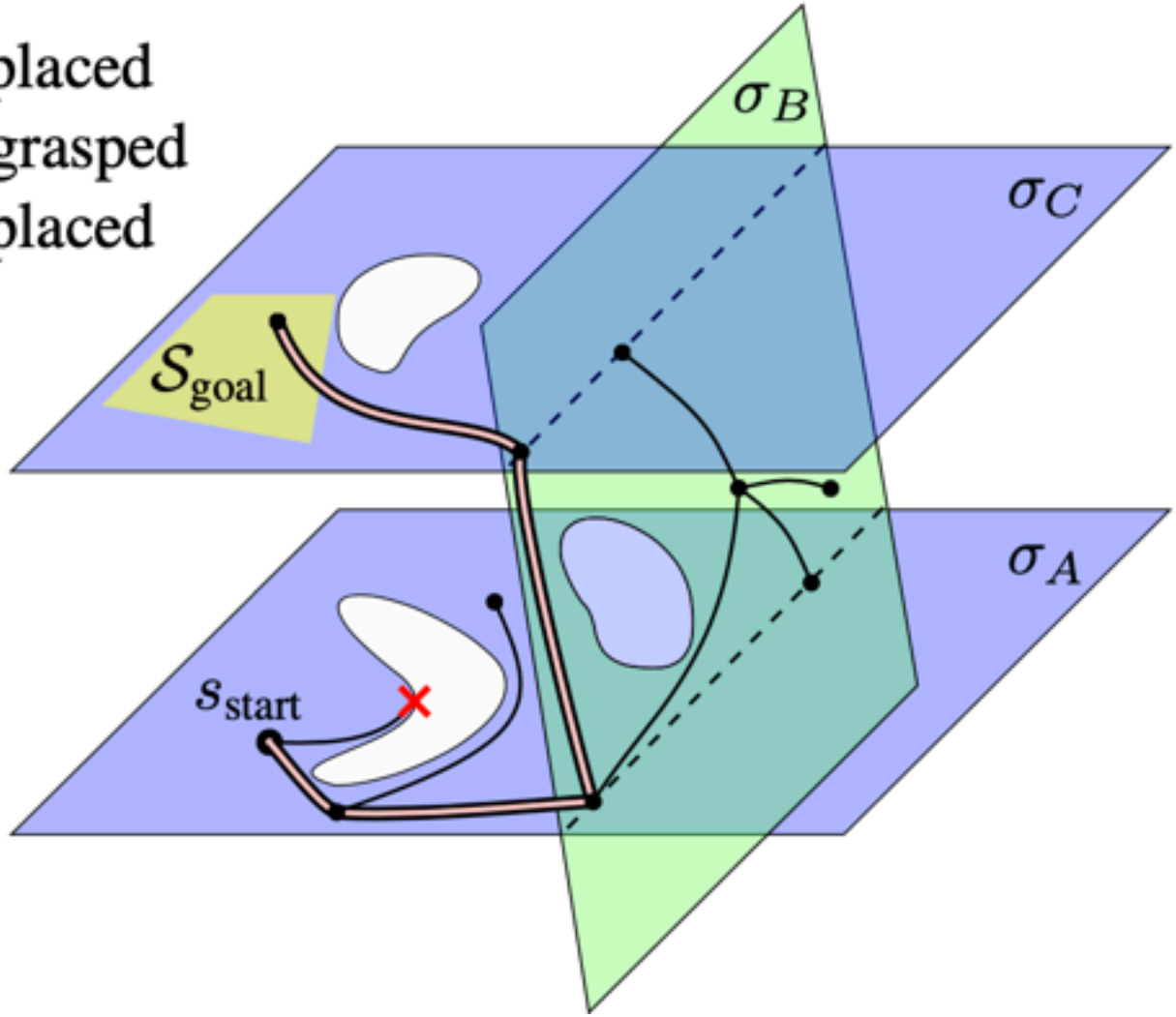
    $s_{\text{new}} \leftarrow \text{simulateController}(s, \pi)$

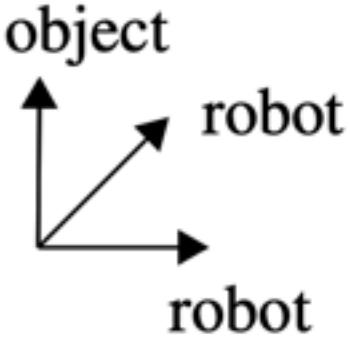
    $N.\text{add}(s_{\text{new}}), \quad E.\text{add}((s, \pi, s_{\text{new}}))$

Explore on neighboring modes

Example

σ_A: object placed
σ_B: object grasped
σ_C: object placed





UC San Diego

Controller Design with State Constraints

- Trajectories $\{q(t), \dot{q}(t)\}_t$ generated by transitionController has to satisfy
  - $f_\sigma(q,t) = 0$, $g_\sigma(q,t) \leq 0$
  - $v_\sigma(q,\dot{q},t) = 0$, $\omega_\sigma(q,\dot{q},t) \leq 0$

- **If we could directly control** f_σ, then we would generate the control signal for f_σ as

$$\ddot{f}_\sigma = -K_p^{f_\sigma} f_\sigma - K_d^{f_\sigma} \dot{f}_\sigma,$$

- But we can only control u (axis acceleration)
-

- **If we could directly control** f_σ, then we would generate the control signal for f_σ as

$$\ddot{f}_\sigma = -K_p^{f_\sigma} f_\sigma - K_d^{f_\sigma} \dot{f}_\sigma,$$

- But we can only control u (axis acceleration). Let us connect $\ddot{f}_\sigma$ and u

- Note that $\ddot{f}_\sigma$ is from the constraint on configurations, thus a function of $\ddot{q}$, and the **affine dynamics model** says $\ddot{q}$ is a function of u:

$$\ddot{q} = a_\sigma(q, \dot{q}, t) + B_\sigma(q, \dot{q}, t)u, \qquad u_{min} \leq u \leq u_{max}$$

- So we can approximate $\ddot{f}_\sigma(u) \approx \ddot{f}_{\sigma,u=0} + \dfrac{\partial \ddot{f}_{\sigma,u=0}}{\partial u} u$

- This gives us the control signal u:

$$\ddot{f}_{\sigma,0} + \frac{\partial \ddot{f}_\sigma}{\partial u} u = -K_p^{f_\sigma} f_\sigma - K_d^{f_\sigma} \dot{f}_\sigma,$$

- Similarly, since we have to satisfy
  - $f_\sigma(q,t) = 0, g_\sigma(q,t) \leq 0$
  - $v_\sigma(q,\dot{q},t) = 0, \omega_\sigma(q,\dot{q},t) \leq 0$
- We will have the full constraints on u

$$\ddot{f}_{\sigma,0} + \frac{\partial \ddot{f}_\sigma}{\partial u} u = -K_p^{f_\sigma} f_\sigma - K_d^{f_\sigma} \dot{f}_\sigma,$$

$$\ddot{g}_{\sigma,0} + \frac{\partial \ddot{g}_\sigma}{\partial u} u \leq -K_p^{g_\sigma} g_\sigma - K_d^{g_\sigma} \dot{g}_\sigma,$$

$$\ddot{f}_{\sigma',0} + \frac{\partial \ddot{f}_{\sigma'}}{\partial u} u = -K_p^{f_{\sigma'}} f_{\sigma'} - K_d^{f_{\sigma'}} \dot{f}_{\sigma'} + \epsilon_{f'},$$

$$\ddot{g}_{\sigma',0} + \frac{\partial \ddot{g}_{\sigma'}}{\partial u} u \leq -K_p^{g_{\sigma'}} g_{\sigma'} - K_d^{g_{\sigma'}} \dot{g}_{\sigma'} + \epsilon_{g'},$$

Some slackness allowed

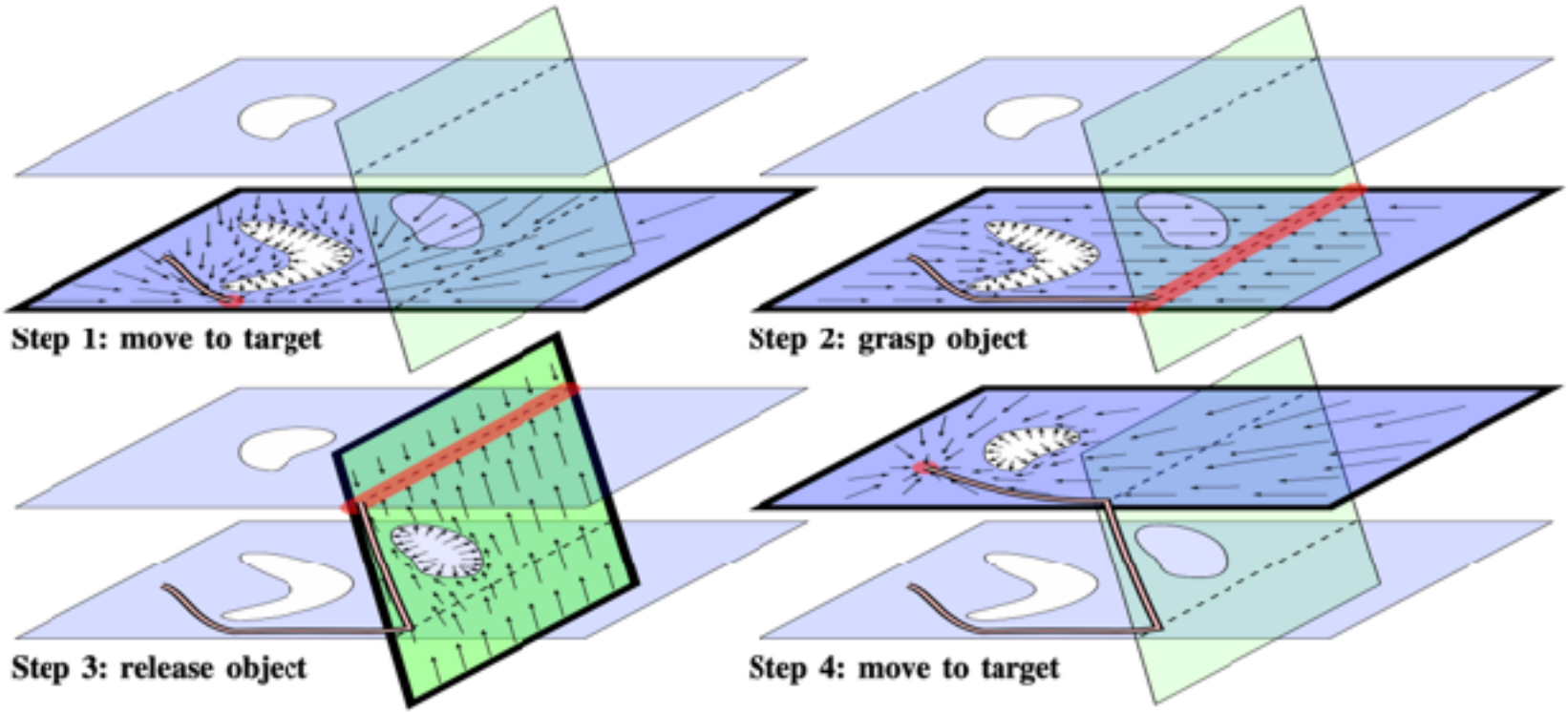
Quadratic Program

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & x^\top H x \\ \text{subject to} \quad & L_A \leq Ax \leq U_A \\ & L \leq x \leq U. \end{aligned}$$

Where $x = [u, \epsilon_{f'}, \epsilon_{g'}]^\top$

- For transitionController, we have the last two constraints involving σ'
- For randomController, we we only need the first two constraints

Example


Step 1: move to target
Step 2: grasp object
Step 3: release object
Step 4: move to target

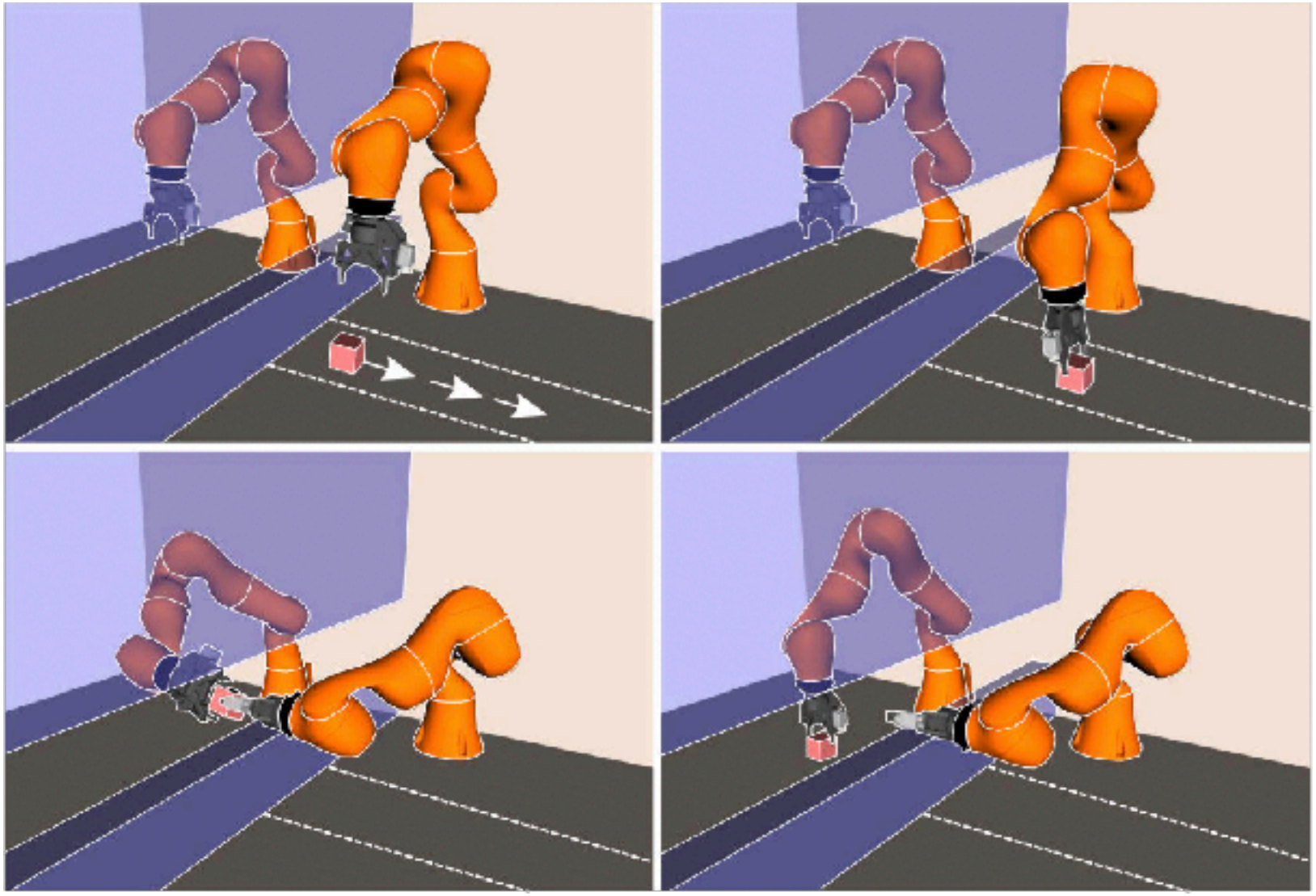
UC San Diego

Experiments

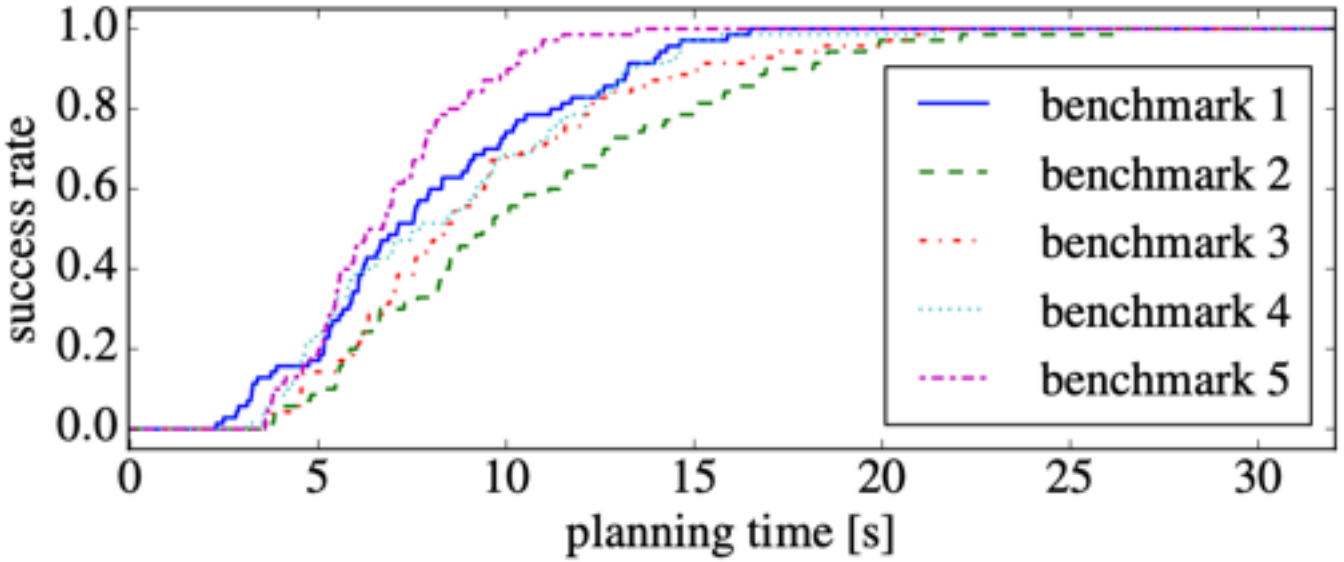
- Five benchmarks based on the two-robot object-passing example

- Benchmark One
  - No obstacles, no moving conveyer belt
- Benchmark Two
  - Only lower wall
- Benchmark Three and Four
  - With lower and higher walls at different height
- Benchmark Five
  - With walls and a moving conveyer belt

Example



Experiment



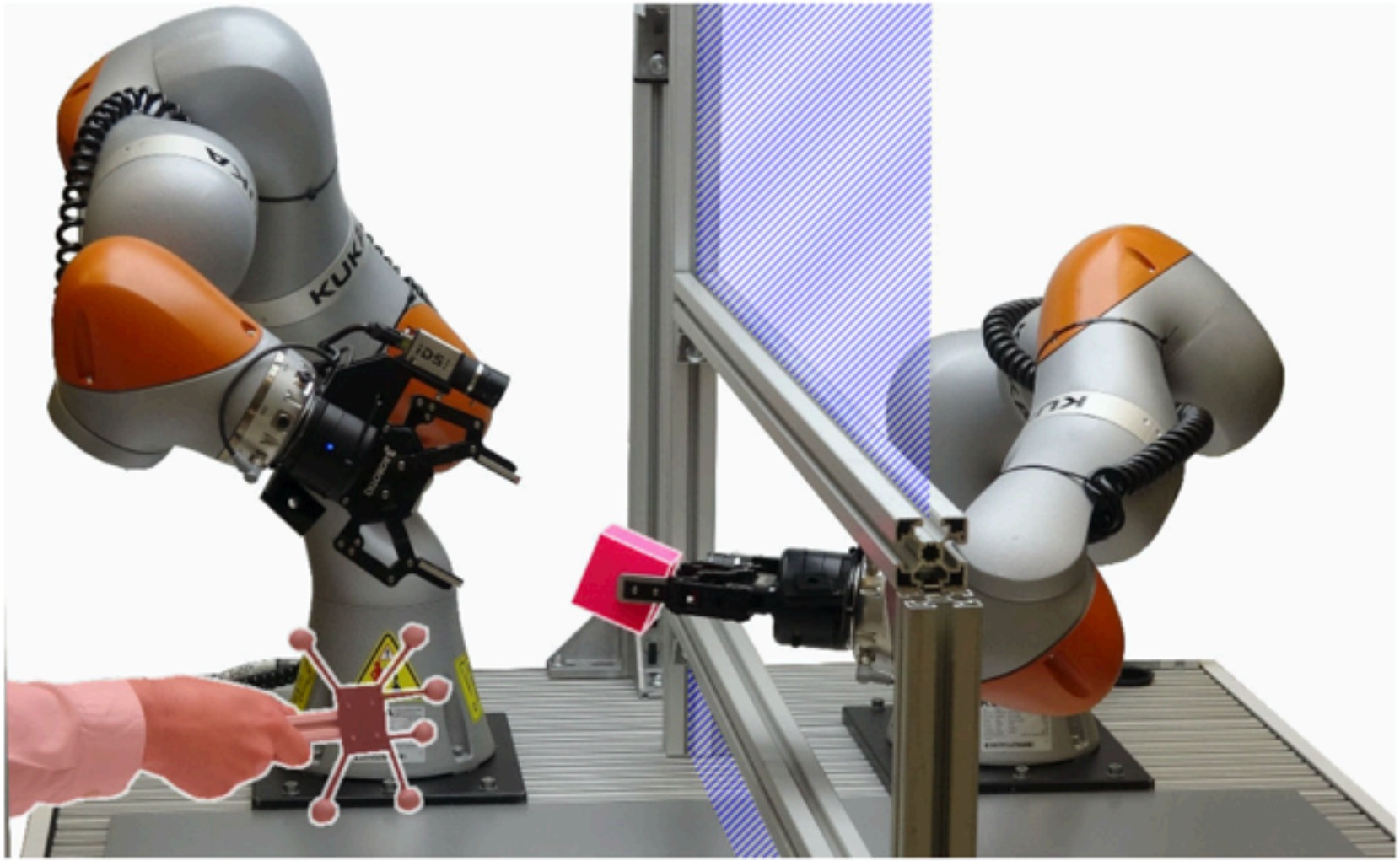
- Both planners(this one and RMR algorithm) were implemented in C++, use multiple threads and were run on a ten-core Intel Xeon E5-2650v3

Experiment

PLANNING AND EXECUTION TIMES

all values in seconds [std. error of the mean]		Benchmark 1	2	3	4	5
proposed planner	planning time	**7.89**	**10.5**	**9.27**	**8.39**	**6.79**
		[0.43]	[0.59]	[0.50]	[0.46]	[0.26]
	execution time	**14.3**	**17.1**	**17.7**	**17.3**	**17.9**
		[0.37]	[0.36]	[0.20]	[0.09]	[0.14]
RMR*	planning time	**6.44**	**8.18**	**21.3**	**>300**	**/**
		[0.67]	[0.79]	[1.54]	/	/
	execution time of first solution	**10.7**	**11.8**	**14.9**	**/**	**/**
		[0.34]	[0.44]	[0.57]	/	/
	min. time at 60s	**5.76**	**5.90**	**7.19**	**/**	**/**

Example In Reality



Conclusion

- Dynamic Constrained Graph
- Kinodynamic Manipulation Planner
- Constraint-Based Controllers as Steering Functions
- Not only a model for robotics simulation but also works in reality. However, no demo is posted.
- Lack of formal guarantee that regarding probabilistic completeness and regions of attraction for the controller sequence.

UC San Diego

Thank you for listening

Any questions?