

# Lecture 15

# Planning - New Frontiers



# Course Logistics

- Quiz 12-13 were posted yesterday and were due before the lecture.
- Project 3 was posted on 10/11 and is due on 10/30 (new).
- Project 4 will be posted on 10/30.



# Where are we in the course?

## Representations

- 1. Transformations
- 2. Rotations & Quaternions

## Manipulation

- 1. Forward Kinematics
- 2. Inverse Kinematics

## Planning

- 1. Path Planning
- 2. Bugs
- 3. Configuration space
- 4. Sampling based planners
- 5. Potential Fields
- 6. Collision Detection

## Motion Control

## Mobile Robotics

# Motion Planning

Sampling-based  
Planning

*PRM, RRT, ...*

Heuristic Discrete  
Search Methods

*Graph, A\*, ...*

Optimization-based  
Planning

*TrajOPT, CHOMP, ...*

# Demo

<https://demonstrations.wolfram.com/ProbabilisticRoadmapMethodForRobotArm/>

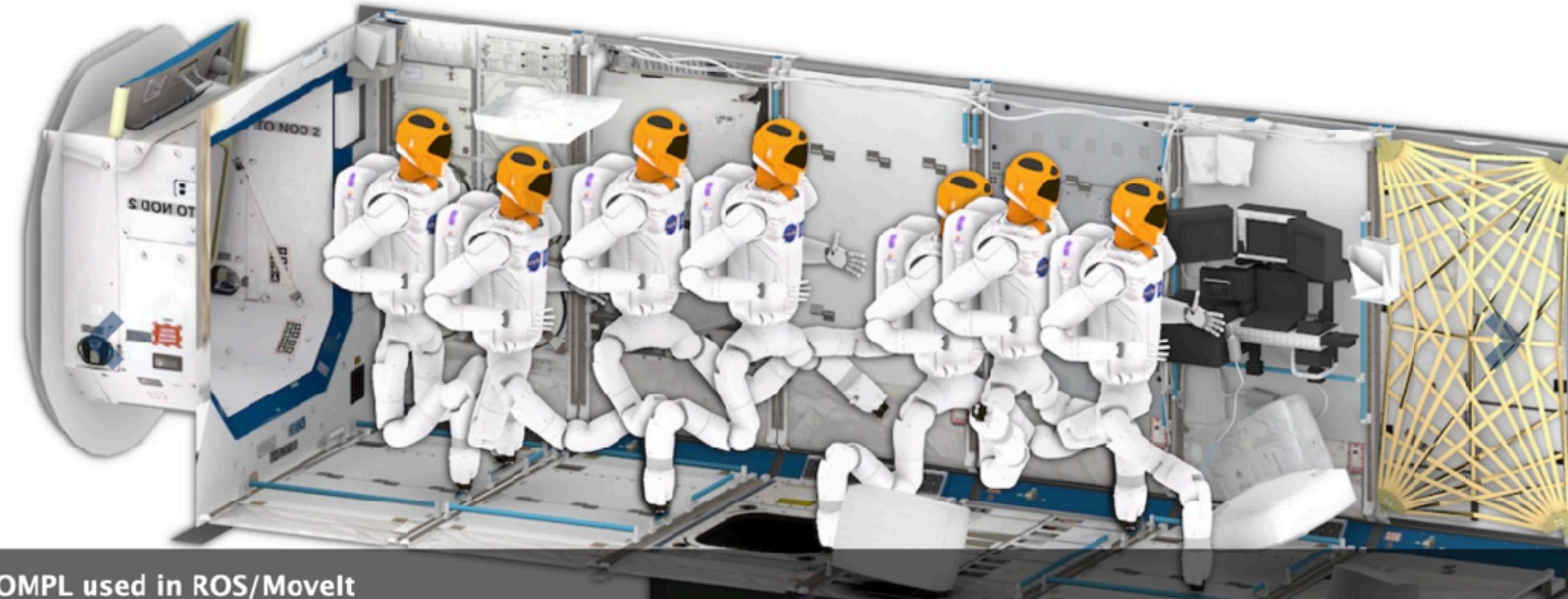


# Motion Planning

OMPL   Download   Documentation ▾   Gallery   OMPL Integrations   Code ▾   Issues   Community ▾   About ▾   Blog

Search

## The Open Motion Planning Library



### OMPL used in ROS/Movelt

OMPL is the default planning library in Movelt and has been used for [many robots](#). Here, OMPL is used to plan footsteps for NASA's Robonaut2 aboard the International Space Station.

OMPL, the Open Motion Planning Library, consists of many state-of-the-art sampling-based motion planning algorithms. OMPL itself does not contain any code related to, e.g., collision checking or visualization. This is a deliberate design choice, so that OMPL is not tied to a particular collision checker or visualization front end. The library is designed so it can be easily integrated into [systems that provide the additional needed components](#).

OMPLapp, the front-end for OMPL, contains a lightweight wrapper for the FCL and PQP collision checkers and a simple GUI based on [PyQt / PySide](#). The graphical front-end can be used for planning motions for rigid bodies and a few vehicle types (first-order and second-order cars, a blimp, and a quadrotor). It relies on the [Assimp](#) library to import a large variety of mesh formats that can be used to represent the robot and its environment.

[Download version 1.6.0](#)

Released: Jan 16, 2023

[Click for citation,  
if you use OMPL in your work](#)

<https://ompl.kavrakilab.org/>

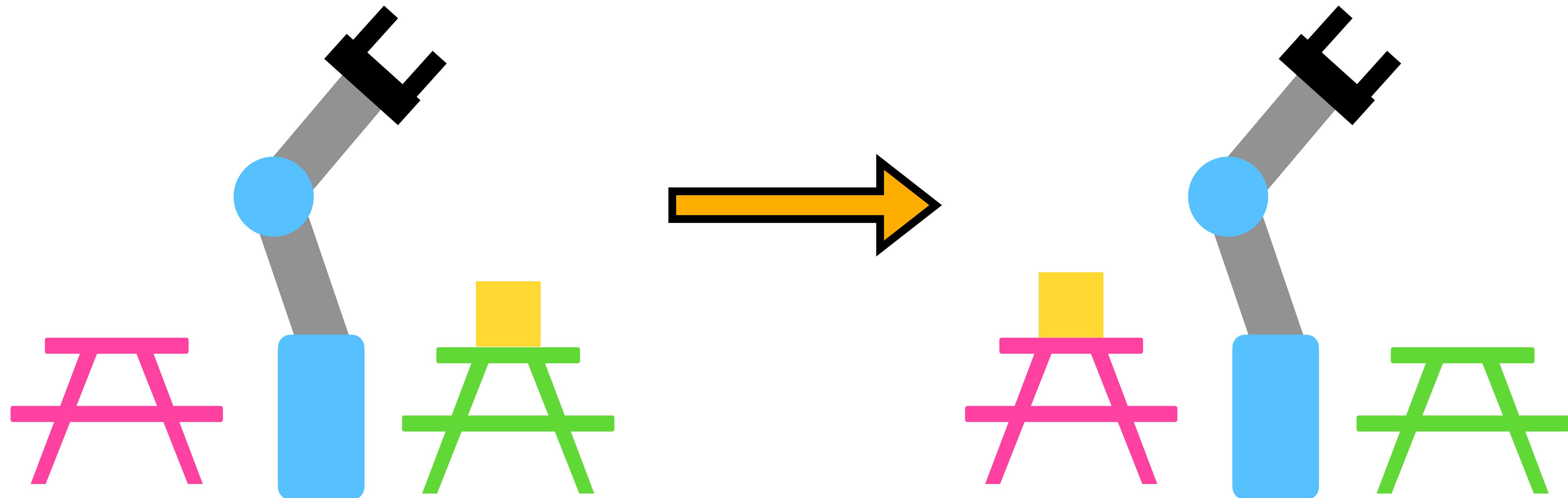


# Demo

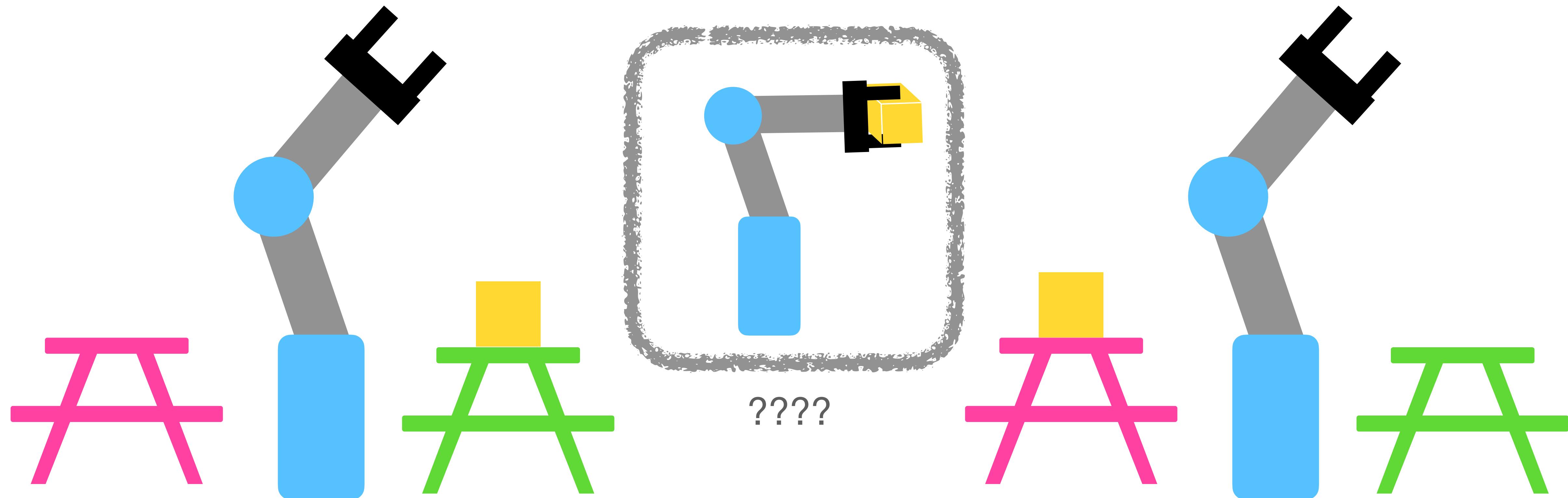
<https://ompl.kavrakilab.org/>



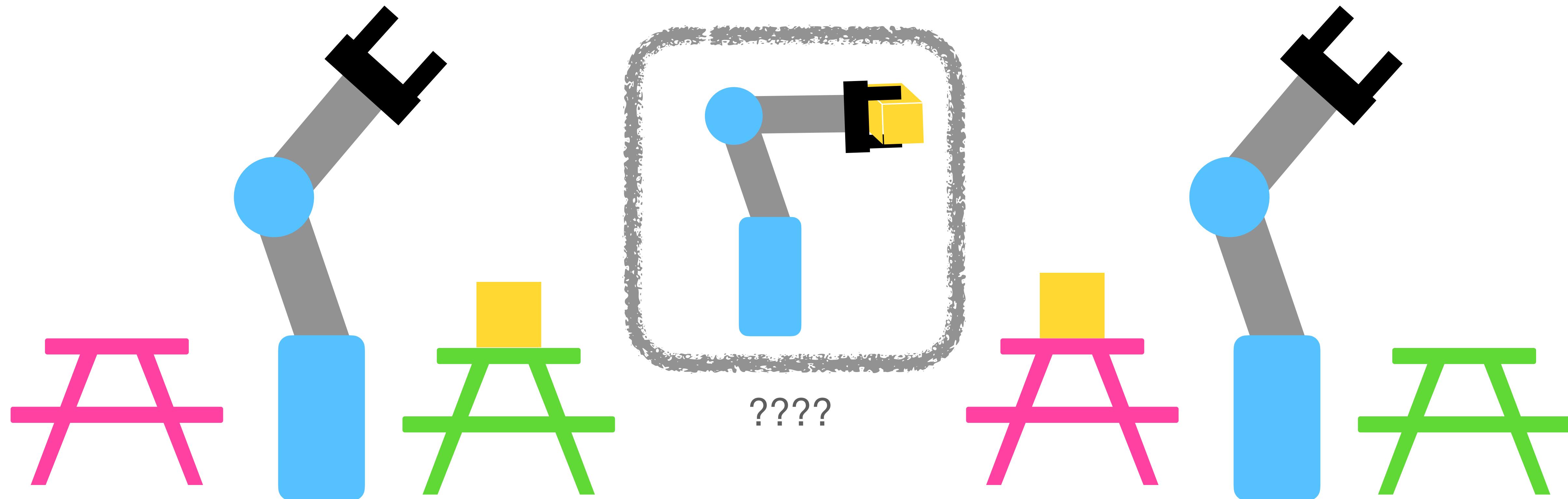
# Is motion planning enough to solve all our problems?



# Is motion planning enough to solve all our problems?



# Think about C-Space during the entire task execution.



# Think about C-Space during the entire task execution.

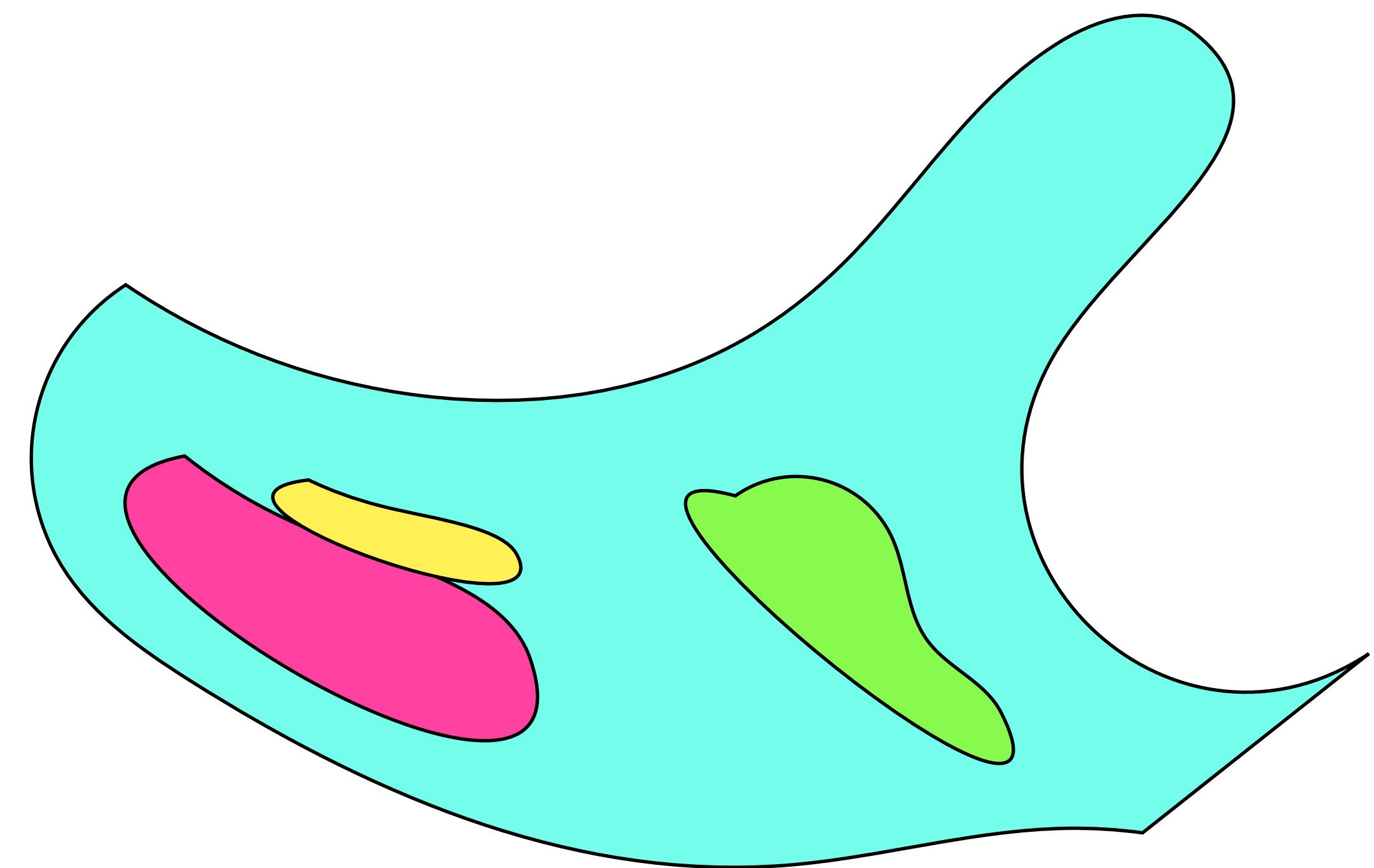
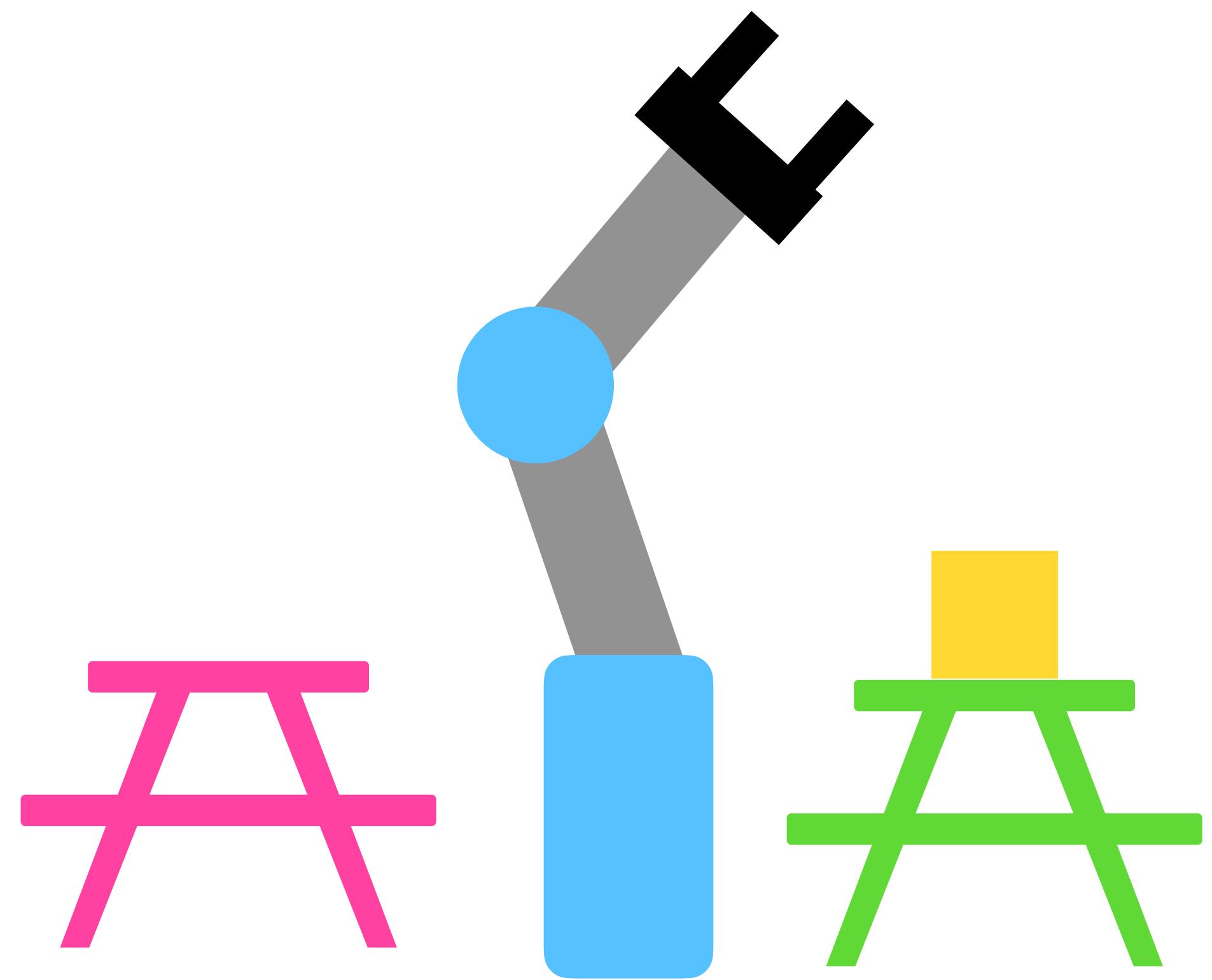


Illustration of a Complex C-space

# Think about C-Space during the entire task execution.

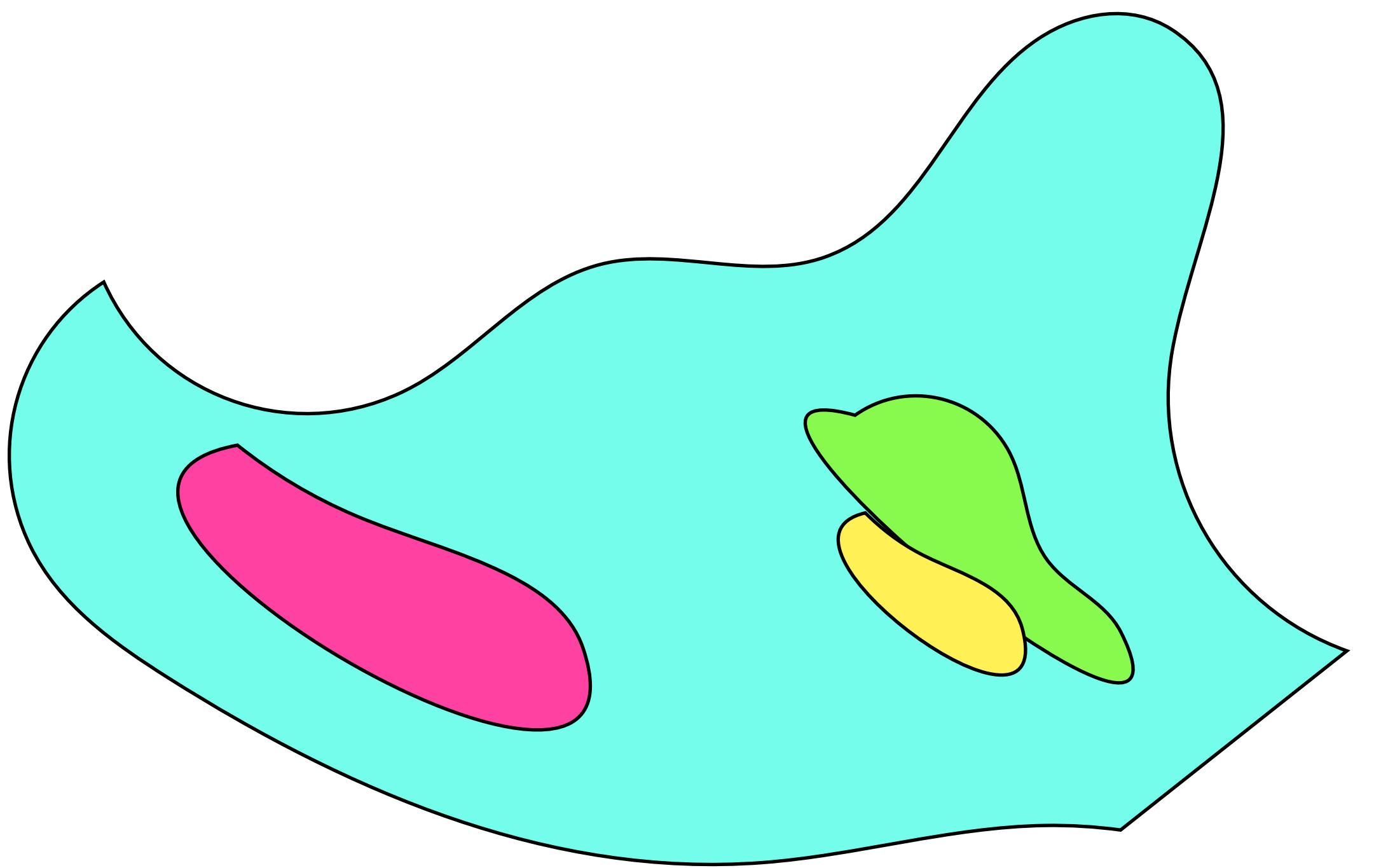
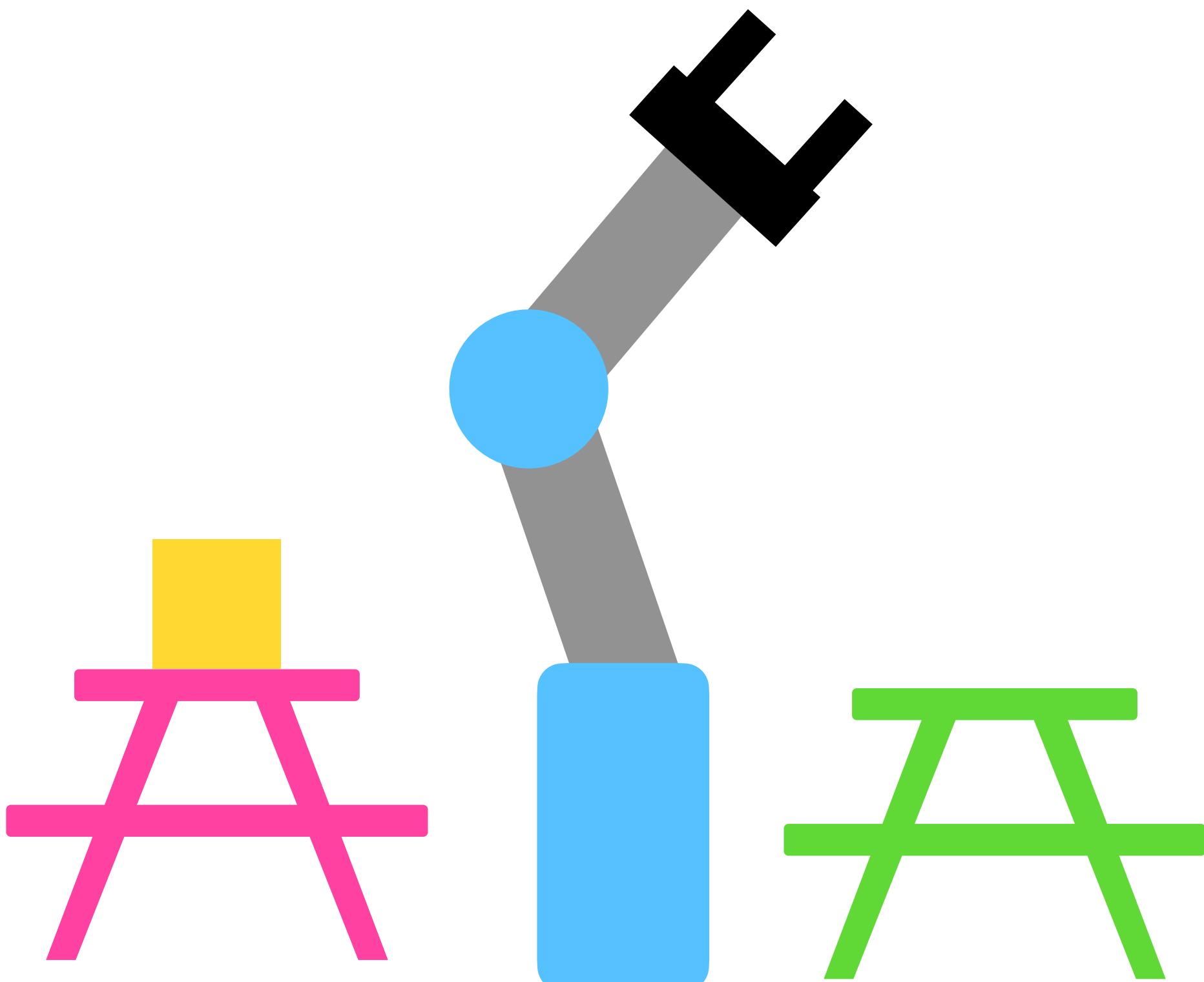
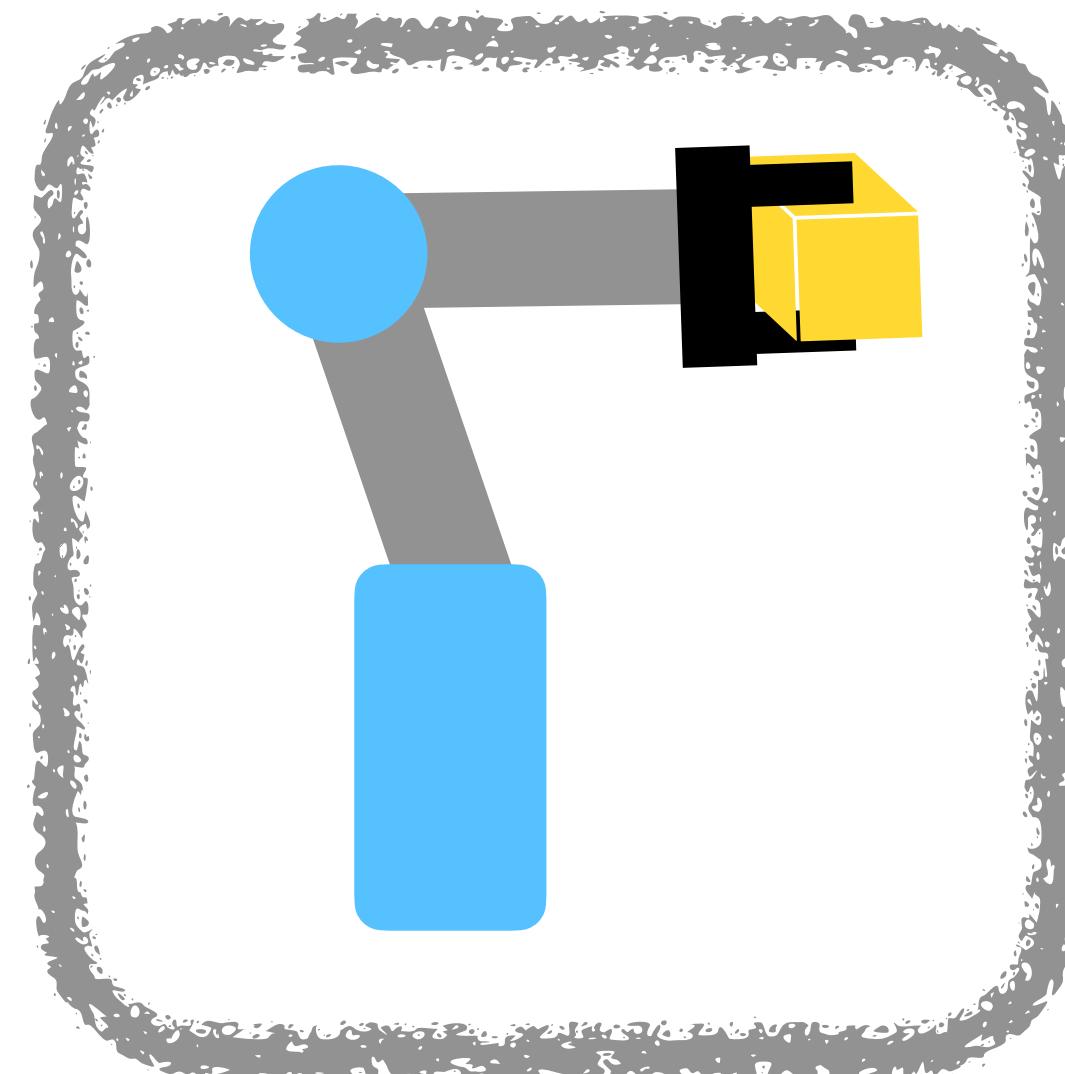


Illustration of a Complex C-space



# Think about C-Space during the entire task execution.



????

What would be a C-space  
after robot grasped the object?

How do we switch between these  
C-spaces?

# Think about C-Space during the entire task execution.



How do we switch between these C-spaces?

Considering we have so many *high-level* actions that robot can perform, how to plan to choose the sequence of *high-level* actions?

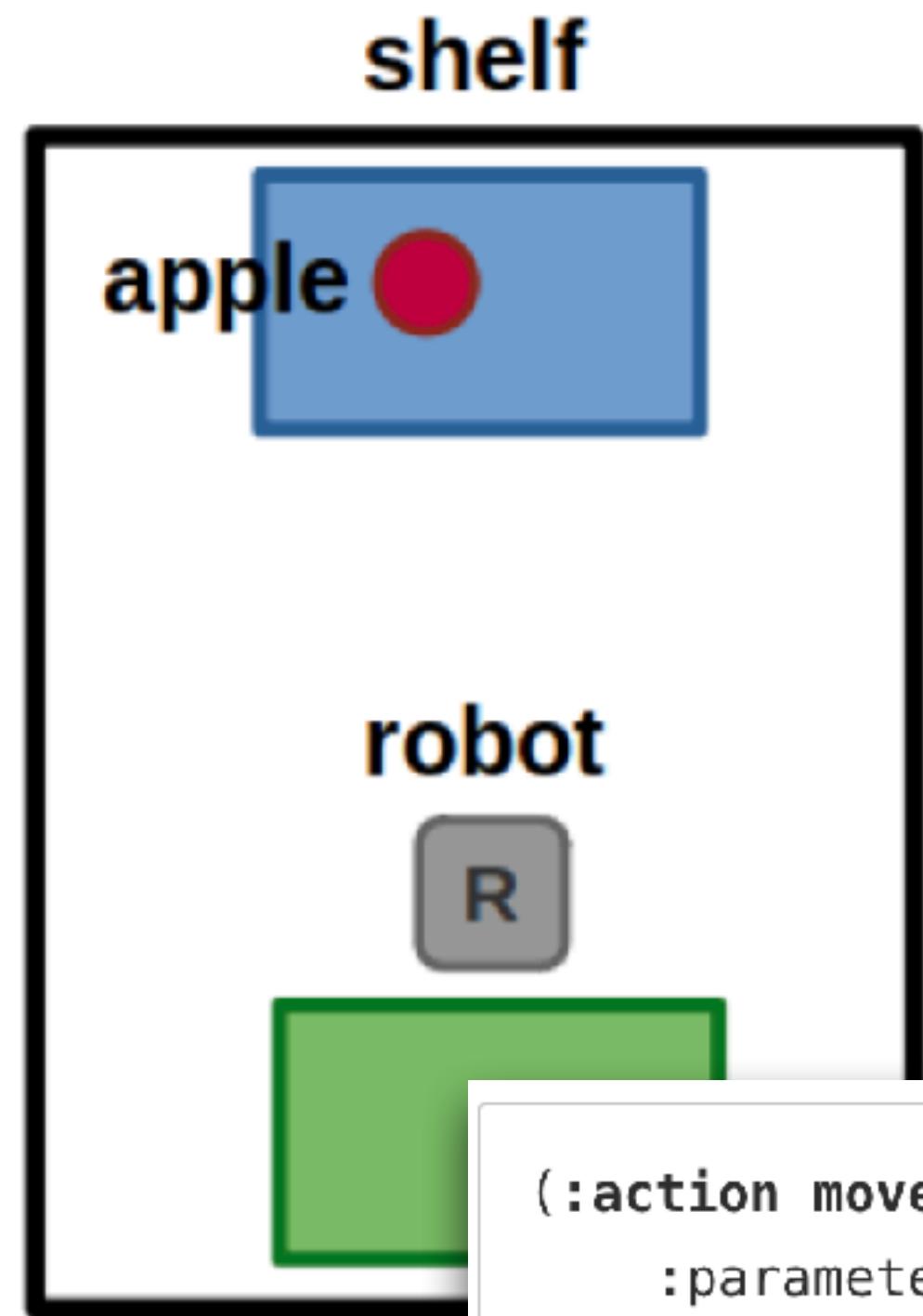
## Task Planning

Robot 'PR2' flips a pancake in a laboratory kitchen of the Institute for Artificial Intelligence (IAI) at the Institute of Informatics and Automation (TzI) of Bremen University in Bremen, Germany. The robot can experimentally do household tasks and is part of a European project.



# Task Planning





## Fetch the apple and put it on the table

1. Move to the *shelf*
2. Pick up the *apple*
3. Move back to the *table*
4. Place the *apple*

```
(:action move
  :parameters (?r ?loc1 ?loc2)
  :precondition (and (Robot ?r)
                      (Location ?loc1)
                      (Location ?loc2)
                      (At ?r ?loc1))
  :effect (and (At ?r ?loc2)
                (not (At ?r ?loc1)))
)
```

Simple task planning example:  
move between a finite  
set of locations  
pick and place objects

- **Domain: The task-agnostic part**

- **Predicates:** (Robot ?r), (Object ?o), (Location ?loc), (At ?r ?loc), (Holding ?r ?o), etc.

- **Actions:** move(?r ?loc1 ?loc2), pick(?r ?o ?loc), place(?r ?o ?loc)

- **Problem: The task-specific part**

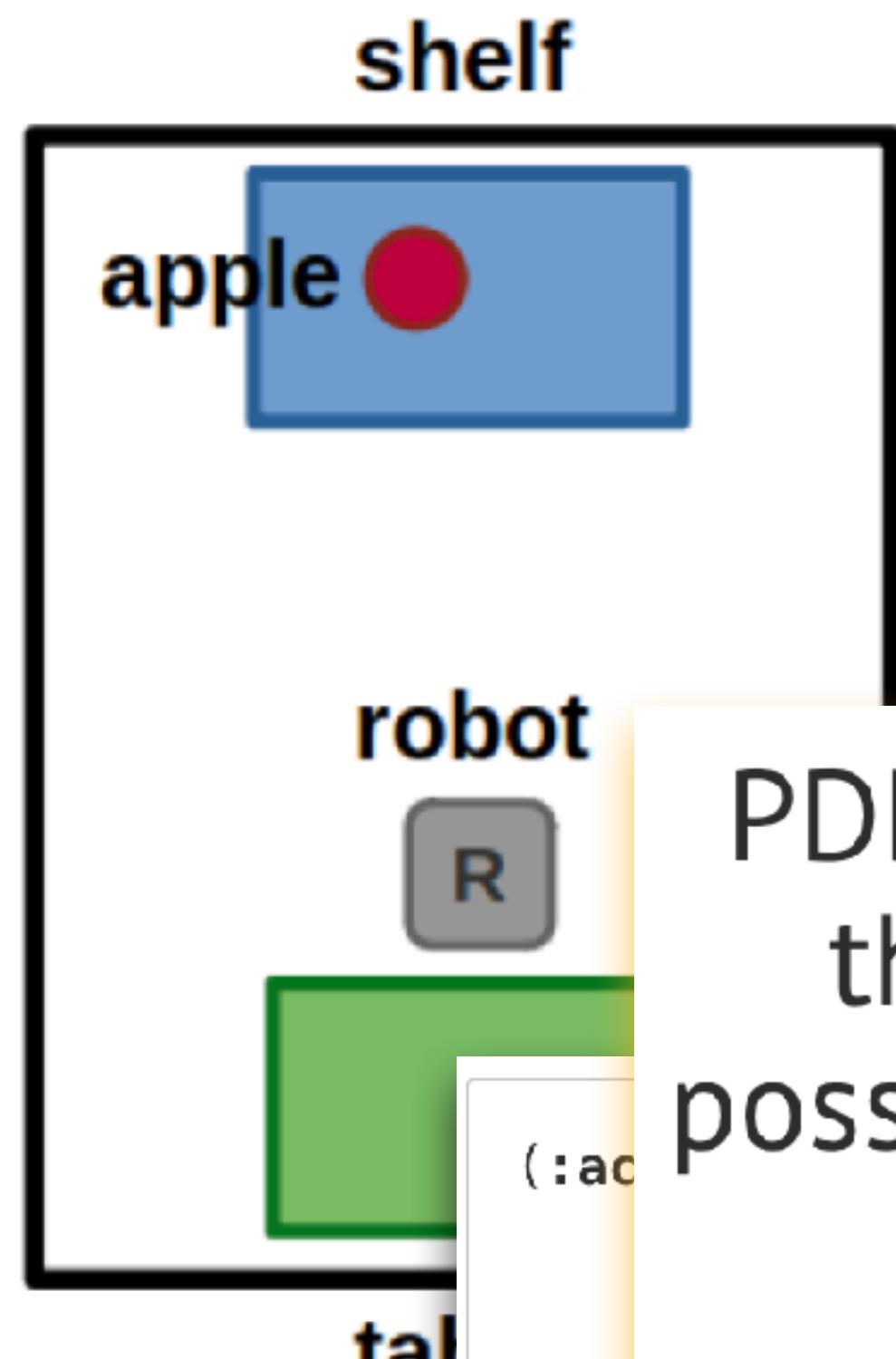
- **Objects:** (Robot robot), (Location shelf), (Location table), (Object apple)

- **Initial state:** (HandEmpty robot), (At robot table), (At apple shelf)

- **Goal specification:** (At apple table)

**on ?loc** is a *unary* predicate, meaning it has one parameter. In our problem, shelf and table are specific location instances, so we say that (on shelf) and (Location shelf) are part of our initial state and do not change over time.

**?loc** is a *binary* predicate, meaning it has two parameters. In our problem, the robot may begin at the table, so we say that (At robot table) is part of our initial state, though it may be negated as the robot moves to another location.



## Fetch the apple and put it on the table

1. Move to the shelf
2. Pick up the apple

PDDL is intended to express the “physics” of a domain, that is, what predicates there are, what actions are possible, what the structure of compound actions is, and what the effects of actions are.

GHALLAB ET AL. (1998), PDDL – THE PLANNING DOMAIN DEFINITION LANGUAGE

```
:effect (and (At ?r ?loc2)
            (not (At ?r ?loc1)))
        )
```

- Domain: The task-agnostic part

- Predicates: (Robot ?r), (Object ?o), (Location ?loc), (At ?r ?loc), (Holding ?r ?o), etc.

- Actions: move(?r ?loc1 ?loc2), pick(?r ?o ?loc), place(?r ?o ?loc)

- Problem: The task-specific part

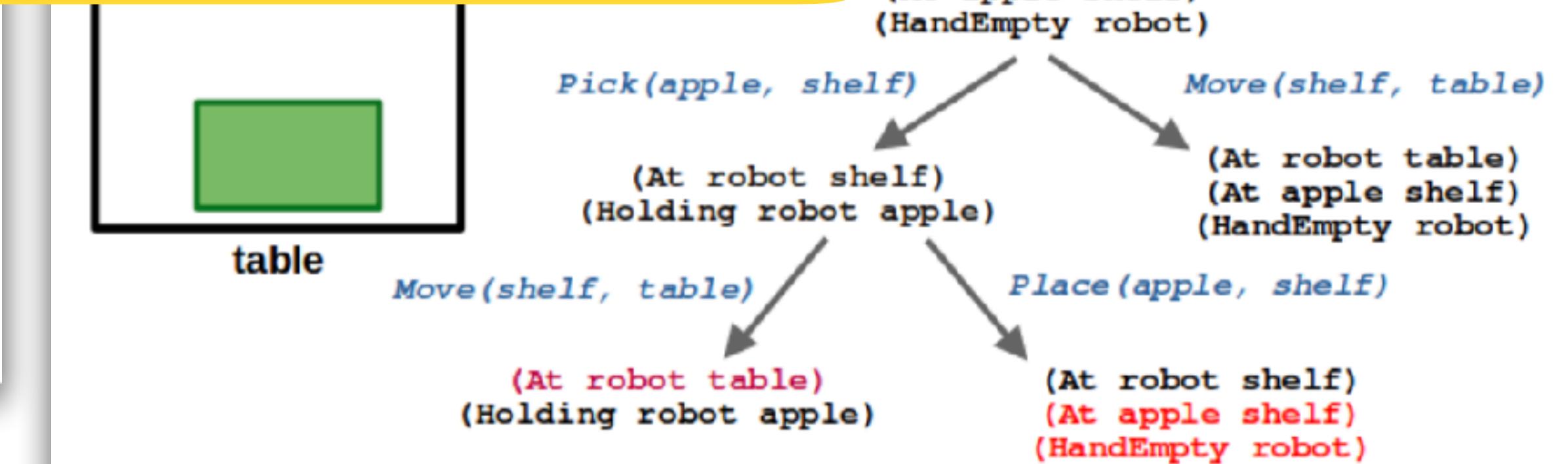
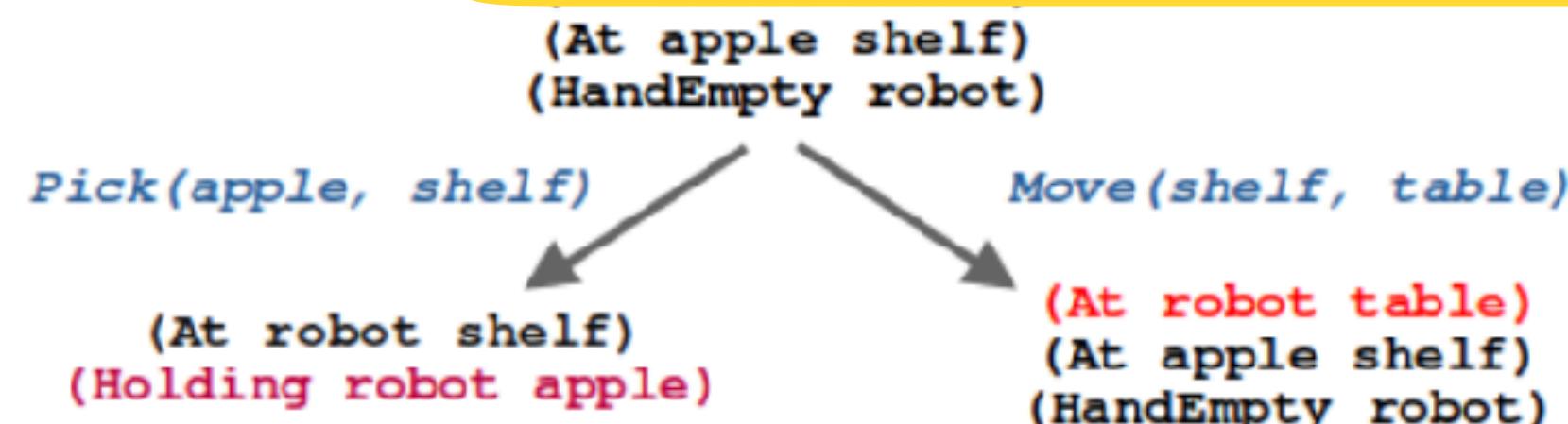
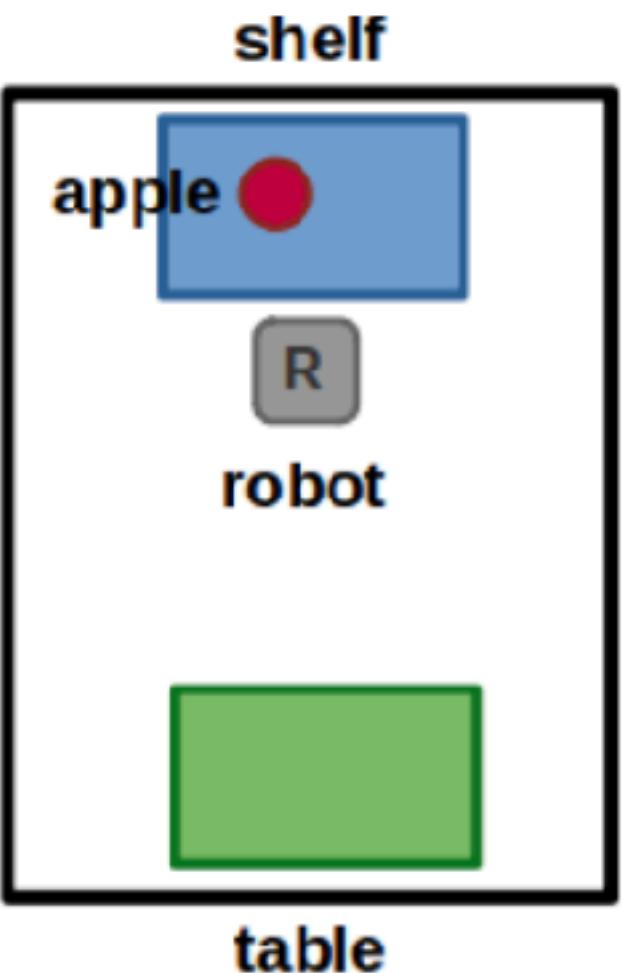
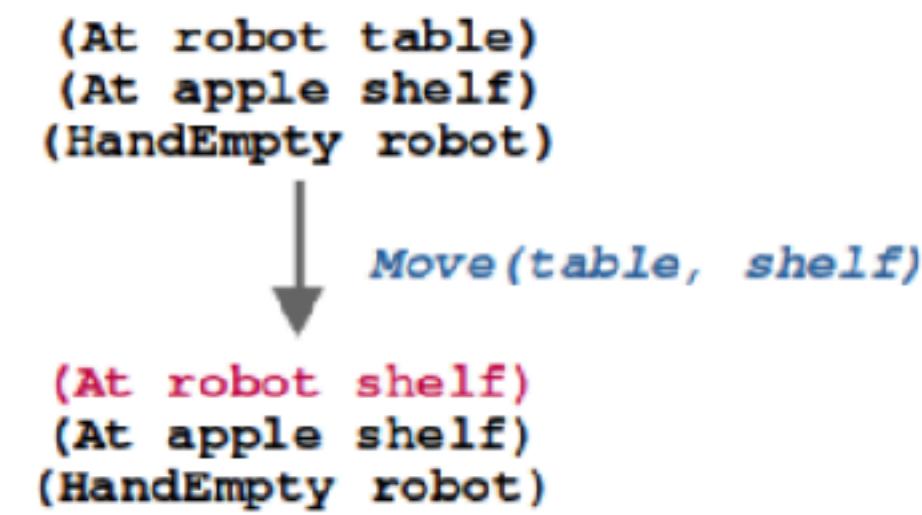
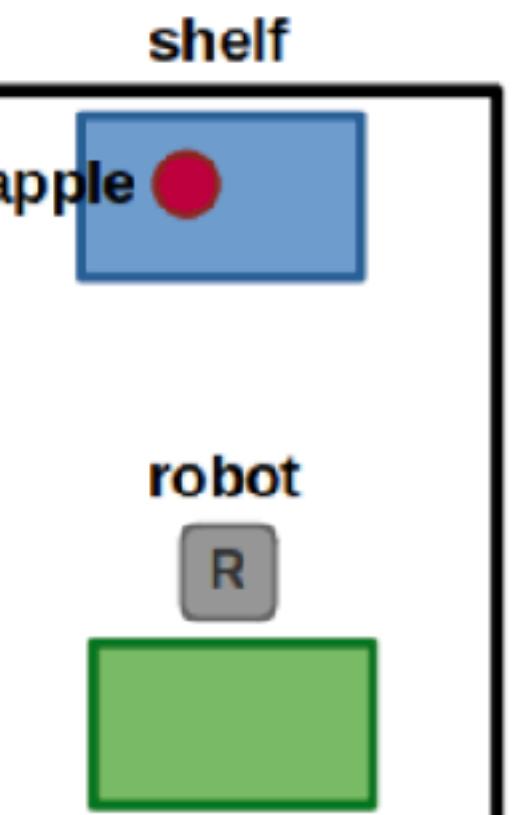
- Objects: (Robot robot), (Location shelf), (Location table), (Object apple)

- Initial state: (HandEmpty robot), (At robot table), (At apple

parameter. In our  
we say that  
our initial state and

parameters. In our

the robot may begin at the table, so we say that (At robot table) is  
initial state, though it may be negated as the robot moves to another



## Discrete Search Algorithms are generally applied

# STRIPS planner

 Search Wikipedia [Create](#)

## Stanford Research Institute Problem Solver

[10 languages](#)[Contents \[hide\]](#)[Article](#) [Talk](#)[Read](#) [Edit](#) [View history](#) [Tools](#)[\(Top\)](#)[Definition](#)[Extensions](#)[A sample STRIPS problem](#)[Complexity](#)[Macro operator](#)[See also](#)[References](#)[Further reading](#)

From Wikipedia, the free encyclopedia

The **Stanford Research Institute Problem Solver**, known by its acronym **STRIPS**, is an [automated planner](#) developed by [Richard Fikes](#) and [Nils Nilsson](#) in 1971 at [SRI International](#).<sup>[1]</sup> The same name was later used to refer to the [formal language](#) of the inputs to this planner. This language is the base for most of the languages for expressing [automated planning](#) problem instances in use today; such languages are commonly known as [action languages](#). This article only describes the language, not the planner.

### Definition [\[edit\]](#)

A STRIPS instance is composed of:

- An initial state;
- The specification of the goal states – situations that the planner is trying to reach;
- A set of actions. For each action, the following are included:
  - preconditions (what must be established before the action is performed);
  - postconditions (what is established after the action is performed).

Mathematically, a STRIPS instance is a quadruple  $\langle P, O, I, G \rangle$ , in which each component has the following meaning:

1.  $P$  is a set of *conditions* (i.e., [propositional variables](#));
2.  $O$  is a set of *operators* (i.e., actions); each operator is itself a quadruple  $\langle \alpha, \beta, \gamma, \delta \rangle$ , each element being a set of conditions. These four sets specify, in order, which conditions must be true for the action to be executable, which ones must be false, which ones are made true by the action and which ones are made false;
3.  $I$  is the initial state, given as the set of conditions that are initially true (all others are assumed false);
4.  $G$  is the specification of the goal state; this is given as a pair  $\langle N, M \rangle$ , which specify which conditions are true and false, respectively, in order for a state to be considered a goal state.

[https://en.wikipedia.org/wiki/Stanford\\_Research\\_Institute\\_Problem\\_Solver](https://en.wikipedia.org/wiki/Stanford_Research_Institute_Problem_Solver)



# Task Planning with LLMs



# Lets ask ChatGPT



I have the following objects on my kitchen counter.



1. Coffee mug.
2. Water Kettle.
3. Water dispenser.
4. Instant Coffee powder in a container.
5. Table spoon.

Additionally, I have the following in the fridge.

1. oat milk,
2. whole milk,
3. chocolate milk.

What steps should I take in order make a cup of coffee with chocolate milk.

# ChatGPT response



To make a cup of coffee with chocolate milk, you can follow these steps:

## Ingredients:

1. Coffee mug
2. Instant coffee powder
3. Chocolate milk (from the fridge)
4. Water (for the coffee)
5. Table spoon
6. Water kettle
7. Water dispenser (for room temperature water)

## Instructions:

### 1. Prepare the Coffee:

- Fill the water kettle with the desired amount of water for your coffee and turn it on to boil. You'll need this hot water to make coffee.
- While waiting for the water to boil, place a table spoon of instant coffee powder into your coffee mug.

### 2. Pour Hot Water:

- Once the water in the kettle has boiled, carefully pour the hot water into the coffee mug with the instant coffee powder. Use the desired amount of hot water to achieve your preferred coffee strength.

### 3. Stir the Coffee:

- Use the table spoon to stir the hot water and instant coffee powder together until the coffee is well mixed and any lumps have dissolved.

### 4. Add Chocolate Milk:

- Open your fridge and take the chocolate milk out.
- Pour the desired amount of chocolate milk into the coffee mug with the coffee. The amount can vary depending on how strong you want your coffee to taste and how much chocolate flavor you prefer.

### 5. Stir Again:

- Use the table spoon to stir the chocolate milk and coffee together until they are well combined.

### 6. Taste and Adjust:

- Taste the coffee with chocolate milk to ensure it's to your liking. If you prefer it sweeter, you can add more chocolate milk. If it's too strong or too sweet, you can add a bit of water to dilute it.

### 7. Enjoy:

- Your cup of coffee with chocolate milk is now ready to enjoy. You can sip it while it's still warm.

Remember to be cautious when handling hot water, and be sure not to overheat your chocolate milk when adding it to the coffee. Adjust the ratios of coffee to chocolate milk to match your taste preferences.

# What are we missing?



# ChatGPT response



To make a cup of coffee with chocolate milk, you can follow these steps:

## Ingredients:

1. Coffee mug
2. Instant coffee powder
3. Chocolate milk (from the fridge)
4. Water (for the coffee)
5. Table spoon
6. Water kettle
7. Water dispenser (for room temperature water)

## Instructions:

### 1. Prepare the Coffee:

- Fill the water kettle with the desired amount of water for your coffee and turn it on to boil. You'll need this hot water to make coffee.
- While waiting for the water to boil, place a table spoon of instant coffee powder into your coffee mug.

### 2. Pour Hot Water:

- Once the water in the kettle has boiled, carefully pour the hot water into the coffee mug with the instant coffee powder. Use the desired amount of hot water to achieve your preferred coffee strength.

How will a robot know what these are?

### 3. Stir the Coffee:

- Use the table spoon to stir the hot water and instant coffee powder together until the coffee is well mixed and any lumps have dissolved.

### 4. Add Chocolate Milk:

- Open your fridge and take the chocolate milk out.
- Pour the desired amount of chocolate milk into the coffee mug with the coffee. The amount can vary depending on how strong you want your coffee to taste and how much chocolate flavor you prefer.

### 5. Stir Again:

- Use the table spoon to stir the chocolate milk and coffee together until they are well combined.

### 6. Taste and Adjust:

- Taste the coffee with chocolate milk to ensure it's to your liking. If you prefer it sweeter, you can add more chocolate milk. If it's too strong or too sweet, you can add a bit of water to dilute it.

### 7. Enjoy:

- Your cup of coffee with chocolate milk is now ready to enjoy. You can sip it while it's still warm.

Remember to be cautious when handling hot water, and be sure not to overheat your chocolate milk when adding it to the coffee. Adjust the ratios of coffee to chocolate milk to match your taste preferences.



# Do As I Can, Not As I Say: Grounding Language in Robotic Affordances

Michael Ahn\* Anthony Brohan\* Noah Brown\* Yevgen Chebotar\* Omar Cortes\* Byron David\* Chelsea Finn\*  
Chuyuan Fu\* Keerthana Gopalakrishnan\* Karol Hausman\* Alex Herzog\* Daniel Ho\* Jasmine Hsu\* Julian Ibarz\*  
Brian Ichter\* Alex Irpan\* Eric Jang\* Rosario Jauregui Ruano\* Kyle Jeffrey\* Sally Jesmonth\* Nikhil Joshi\*  
Ryan Julian\* Dmitry Kalashnikov\* Yuheng Kuang\* Kuang-Huei Lee\* Sergey Levine\* Yao Lu\* Linda Luu\* Carolina Parada\*  
Peter Pastor\* Jornell Quiambao\* Kanishka Rao\* Jarek Rettinghouse\* Diego Reyes\* Pierre Sermanet\* Nicolas Sievers\*  
Clayton Tan\* Alexander Toshev\* Vincent Vanhoucke\* Fei Xia\* Ted Xiao\* Peng Xu\* Sichun Xu\* Mengyuan Yan\* Andy Zeng\*



Robotics at Google



Everyday Robots

\* Authors listed in alphabetical order (see paper appendix for contribution statement).



Paper



Video



Blogpost



Code



Demo

## What's New

- [8/16/2022] We integrated SayCan with [Pathways Language Model \(PaLM\)](#), and updated the results. We also added [new capabilities](#) including drawer manipulation, chain of thought prompting and multilingual instructions. You can see all the new results in the updated [paper](#).
- [8/16/2022] Our updated results show that SayCan combined with the improved language model (PaLM), which we refer to as PaLM-SayCan, improves the **robotics performance** of the entire system compared to a previous LLM (FLAN). PaLM-SayCan chooses the correct sequence of skills 84% of the time and executes them successfully 74% of the time, reducing errors by a half compared to FLAN. This is particularly exciting because it represents the first time we can see how an improvement in language models translates to a similar improvement in robotics.
- [8/16/2022] We [open-sourced](#) a version of SayCan on a simulated tabletop environment.
- [4/4/2022] Initial release of SayCan.

<https://say-can.github.io/>

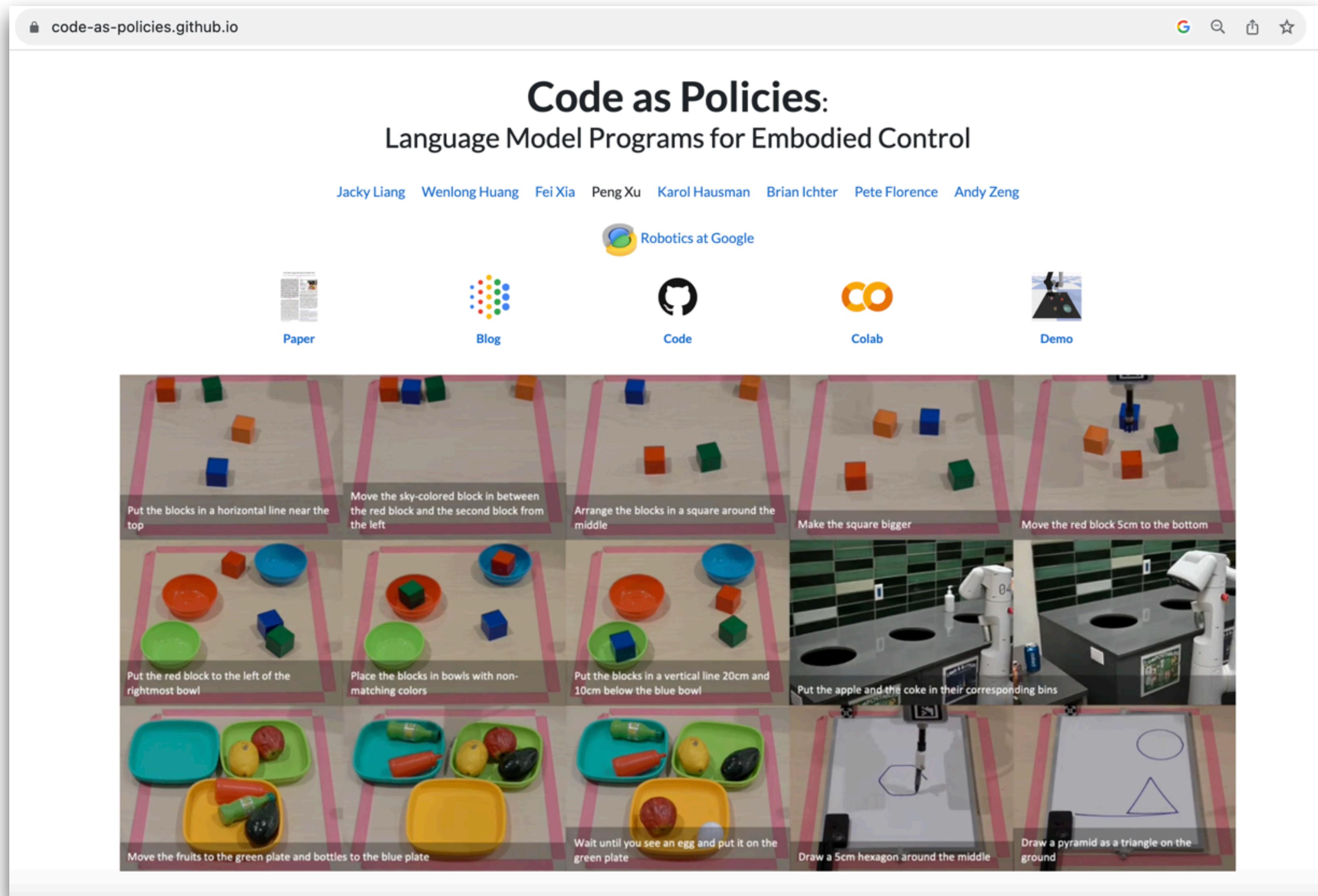
# 4x speed



User input: I just worked out, can you bring me a drink and a snack to recover?

Robot: I would 1. find a water bottle, 2. pick up the water bottle  
3. bring it to you, 4. \_\_





<https://code-as-policies.github.io/>

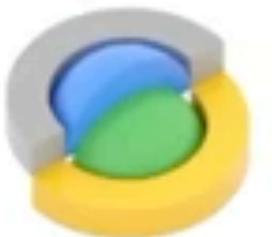


# Code as Policies: Language Model Programs for Embodied Control

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, Andy Zeng

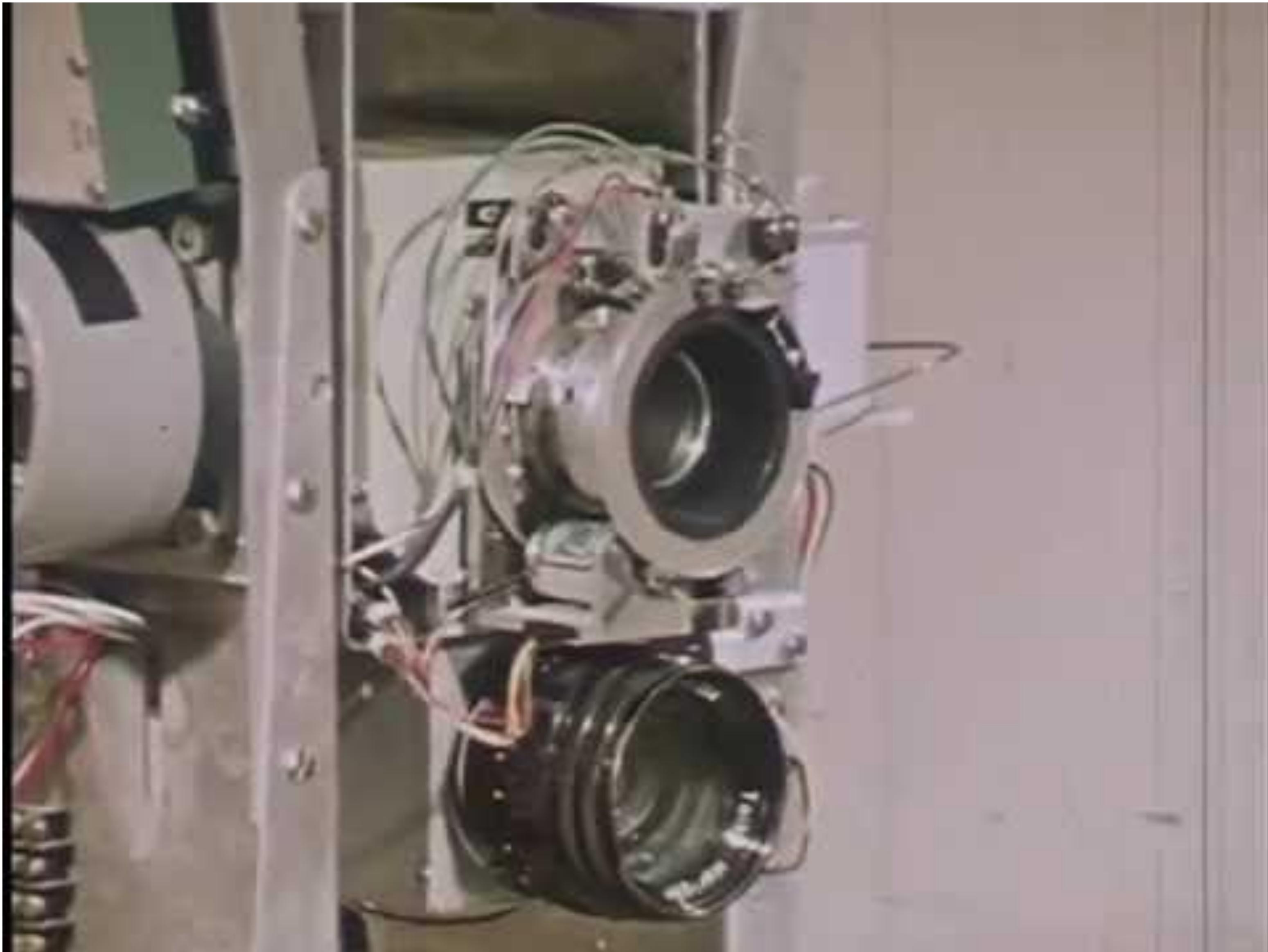
This video is voice-narrated

[code-as-policies.github.io](https://code-as-policies.github.io)



Robotics at Google

# Task and Motion Planning



From 1966 through 1972, the Artificial Intelligence Center at SRI International (then Stanford Research Institute) conducted research on a mobile robot system nicknamed “Shakey.” Endowed with a limited ability to perceive and model its environment, Shakey could perform tasks that required planning, route-finding, and the rearranging of simple objects.

<https://www.youtube.com/watch?v=GmU7SimFkpU>

# Task and Motion Planning

## Integrated Task and Motion Planning

Caelan Reed Garrett<sup>1</sup>, Rohan Chitnis<sup>1</sup>, Rachel Holladay<sup>1</sup>, Beomjoon Kim<sup>1</sup>, Tom Silver<sup>1</sup>, Leslie Pack Kaelbling<sup>1</sup> and Tomás Lozano-Pérez<sup>1</sup>

<sup>1</sup>CSAIL, MIT, Cambridge, USA, 02139; email: caelan@csail.mit.edu

Annu. Rev. Control Robot. Auton. Syst.  
2021. 2021. 4:1–30  
Copyright © 2021 by Annual Reviews.  
All rights reserved.

### Keywords

task and motion planning, robotics, automated planning, motion planning, manipulation planning

### Abstract

The problem of planning for a robot that operates in environments containing a large number of objects, taking actions to move itself through the world as well as to change the state of the objects, is known as *task and motion planning* (TAMP). TAMP problems contain elements of discrete task planning, discrete-continuous mathematical programming, and continuous motion planning, and thus cannot be effectively addressed by any of these fields directly. In this paper, we define a class of TAMP problems and survey algorithms for solving them, characterizing the solution methods in terms of their strategies for solving the continuous-space subproblems and their techniques for integrating the discrete and continuous components of the search.

	Pre-discretized	Sampling	Optimization
Satisfaction First	Ferrer-Mestres <sup>*</sup> (84, 85)	Simón <sup>†</sup> (22) Hausser <sup>†</sup> (13, 29, 14) Garrett <sup>*</sup> (86, 21) Krontiris <sup>†</sup> (87, 88) Akbari <sup>*</sup> (89) Vega-Brown <sup>†</sup> (90)	
Interleaved	Dornhege <sup>*</sup> (62, 63, 91) Gaschler <sup>*</sup> (92, 93, 94) Colledanchise <sup>*</sup> (95)	Gravot <sup>*</sup> (96, 97) Stilman <sup>†</sup> (23, 98, 99) Plaku <sup>†</sup> (100) Kaelbling <sup>*</sup> (101, 102) Barry <sup>†</sup> (103, 30, 104) Garrett <sup>*</sup> (70, 71) Thomason <sup>*</sup> (105) Kim <sup>*</sup> (106, 107) Kingston <sup>†</sup> (108)	Fernandez-Gonzalez <sup>*</sup> (109)
Sequence First	Nilsson <sup>*</sup> (2) Erdem <sup>*</sup> (74, 75) Lagriffoul <sup>*</sup> (65, 66, 67) Pandey <sup>*</sup> (110, 111) Lozano-Pérez <sup>*</sup> (112) Dantam <sup>*</sup> (77, 78, 79) Lo <sup>*</sup> (113)	Wolfe <sup>*</sup> (114) Srivastava <sup>*</sup> (76, 60) Garrett <sup>*</sup> (55, 73)	Toussaint <sup>*</sup> (61, 68, 69) Shoukry <sup>*</sup> (81, 82, 83) Hadfield-Menell <sup>*</sup> (115)

Table 1: A table that categorizes MMMP and TAMP approaches, based on how they solve HCS-SPS and how they integrate with constraint satisfaction with action sequencing. Approaches for MMMP are designated with <sup>†</sup>, and approaches for TAMP are designated with <sup>\*</sup>. Each table cell is listed chronologically.

Garrett, Caelan Reed, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. "Integrated task and motion planning." *Annual review of control, robotics, and autonomous systems* 4 (2021): 265-293.



# Next Lecture

## Motion Control - I

