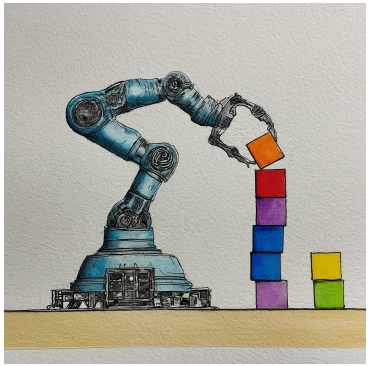
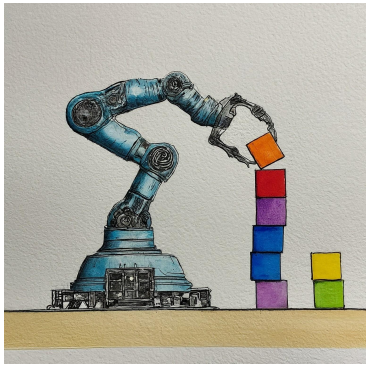
Learning for Integrated Task and Motion Planning

2025 AAAI Bridge Program



Content

- About the program
- Schedule overview
- An introduction to integrated task and motion planning



Organizing Team

Sarah Keren

Collaborative AI and Robotics

TECHNION

Brian Williams

Model-based Embedded and Robotic Systems

MIT CSAIL

Michael Posa

Dynamic Autonomy and Intelligent Robotics Lab

Penn Engineering
GRASP LABORATORY
General Robotics, Automation, Sensing & Perception Lab

Our github links

https://github.com/CLAIR-LAB-TECHNION/AAAI_25_Bridge_TMP

https://github.com/CLAIR-LAB-TECHNION/CLAIR-TMP-Tutorials

Schedule

Day 1

Time	Session
9:00 AM - 10:30 AM	Intro + Tutorial 1 - Sarah Keren (Technion)
10:30 AM - 11:00 AM	Break
11:00 AM - 12:30 PM	Brian Williams (MIT)
12:30 PM - 2:00 PM	Lunch (on your own; no sponsored lunch provided)
2:00 PM - 3:30 PM	Jiayuan Mao (MIT) + Tutorial 2 -Sarah Keren (Technion)
3:30 PM - 4:00 PM	Break
4:00 PM - 5:30 PM	Michael Posa (UPenn) + Laser Talks

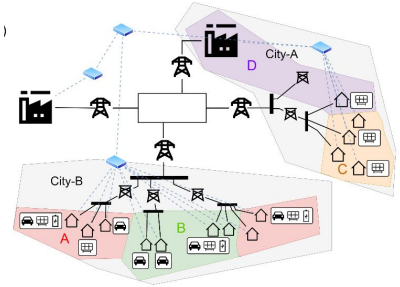
Day 2

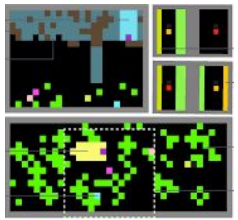
Time	Session
9:00 AM - 10:30 AM	Tutorial 3 - Sarah Keren (Technion)
10:30 AM - 11:00 AM	Break
11:00 AM - 12:30 PM	David Held (CMU)
12:30 PM - 2:00 PM	Lunch (no sponsored lunch provided)
2:00 PM - 3:30 PM	Peter Stone (University of Texas at Austin and Sony AI)
3:30 PM - 4:00 PM	Break
4:00 PM - 5:30 PM	Panel: Peter Stone, David Held, Michael Posa, Brian Williams, Rachel Holladay (UPenn), Sarah Keren

A Very Brief Intro to Task and Motion Planning

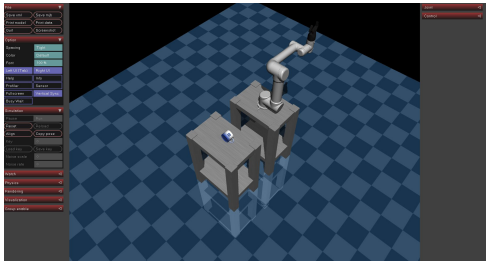
Sarah Keren

TECHNION | The Henry and Marilyn Taub Faculty of Computer Science





Collaborative AI and Robotics

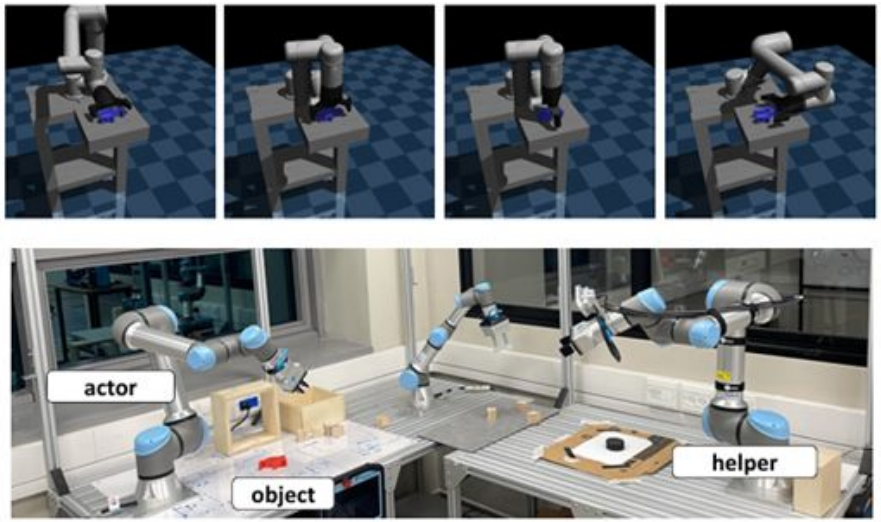


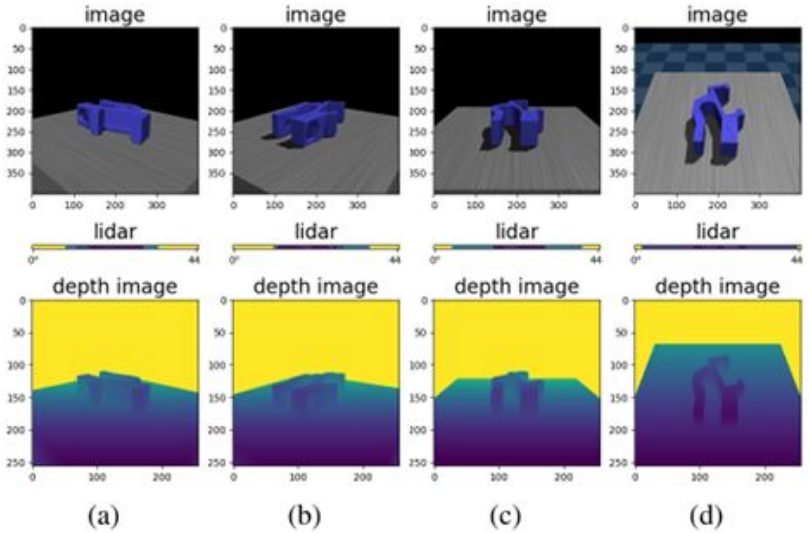
Sarah Keren

TECHNION | The Henry and Marilyn Taub Faculty of Computer Science

One of our projects

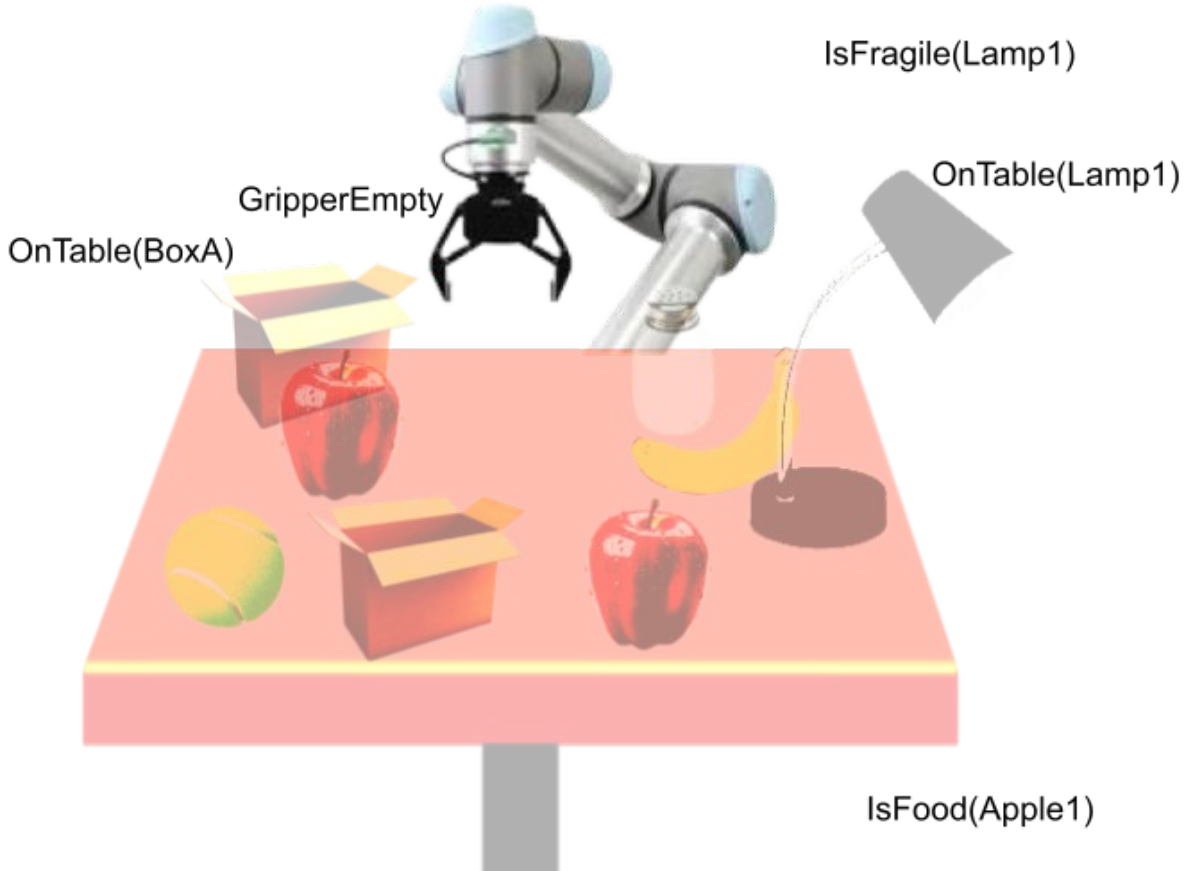
Value of Assistance for long term decision making under uncertainty

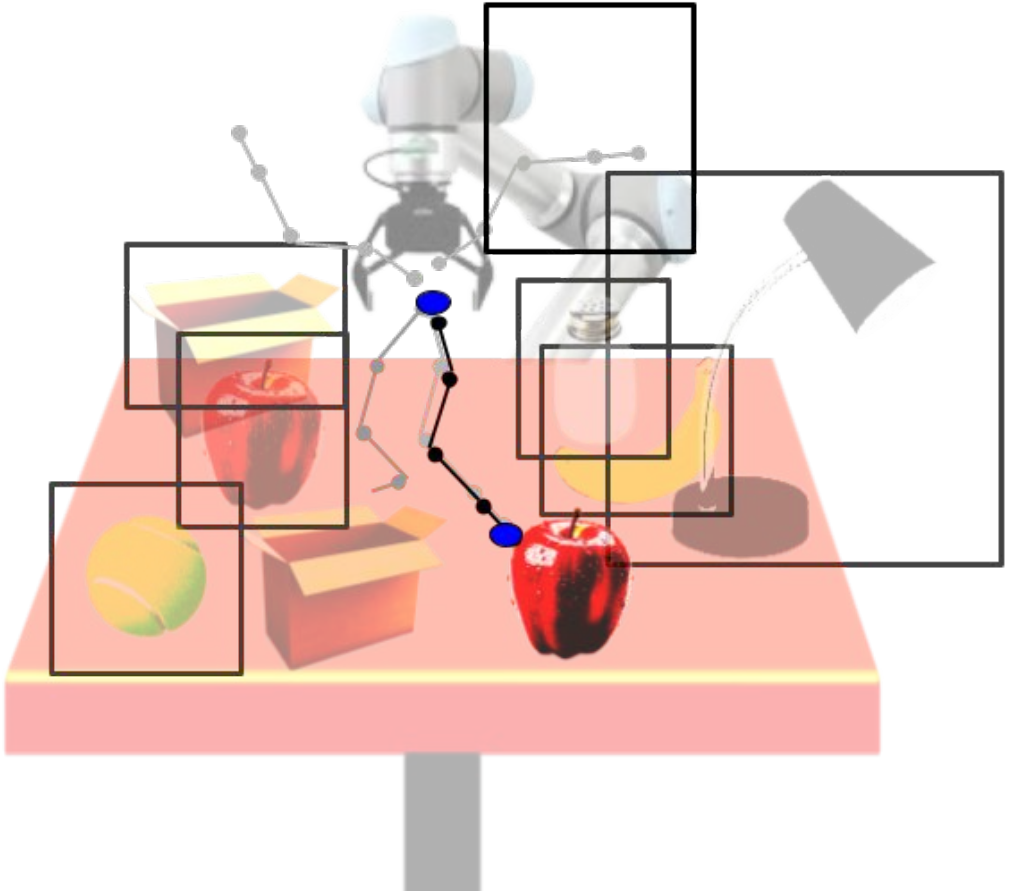


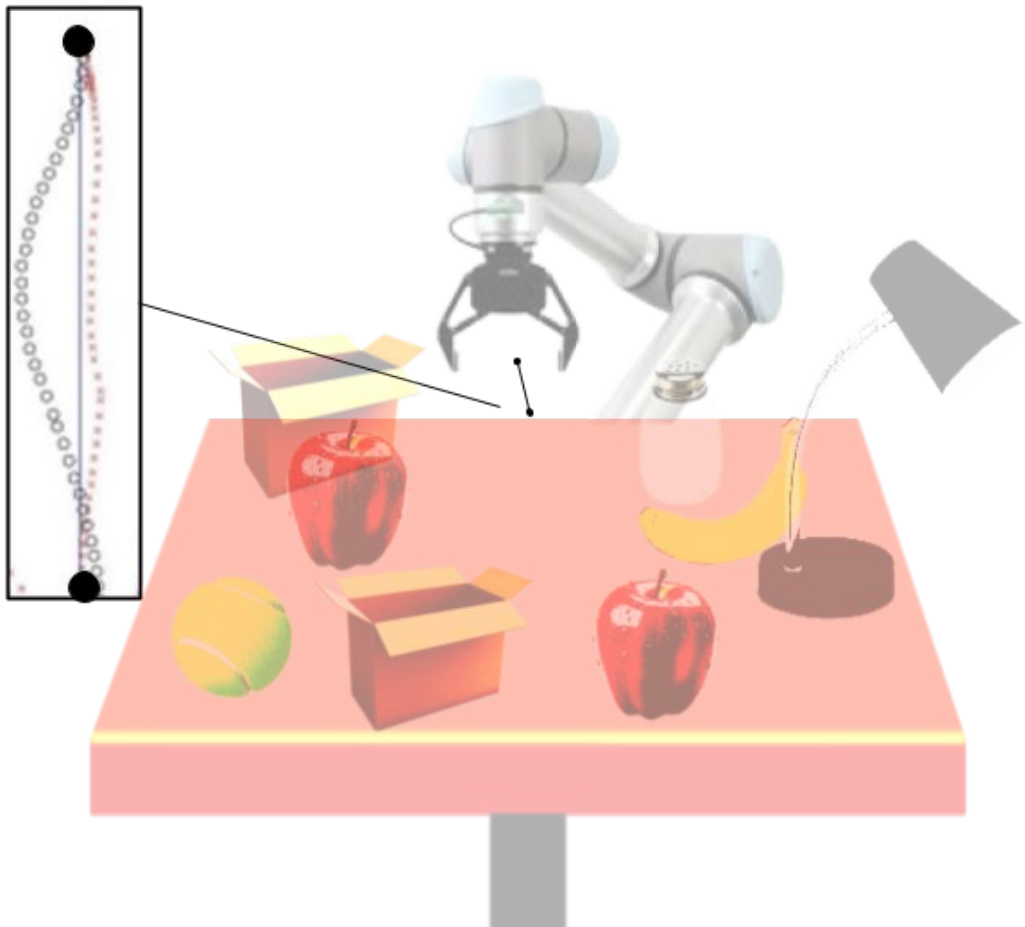


Example

“I am hungry”


IsFragile(Lamp1)
OnTable(Lamp1)
GripperEmpty
OnTable(BoxA)
IsFood(Apple1)





Outline

Motivation for integrating task and motion planning (and control)

Components of a robotic agent

Task planning and its limitations

Motion planning and its limitations

Integrating task and motion planning

Let’s (of course) start with a blocks world example

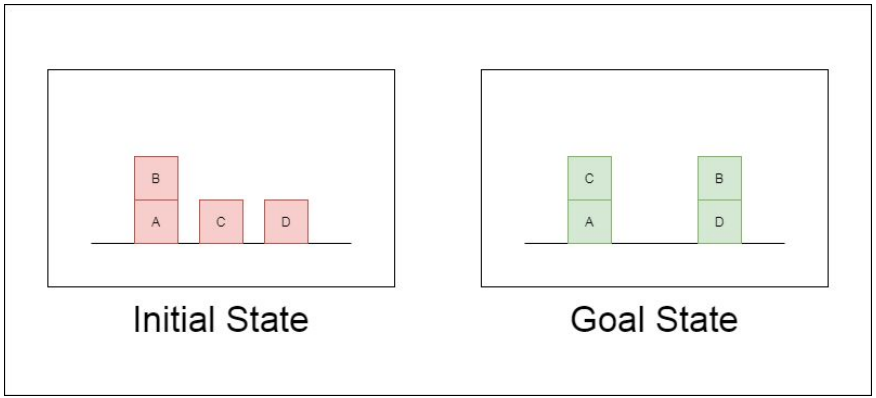
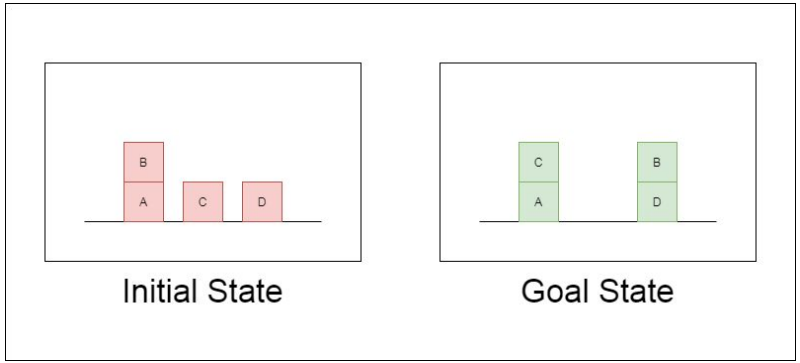


Image from https://apoorvdixit619.medium.com/goal-stack-planning-for-blocks-world-problem-41779d090f29

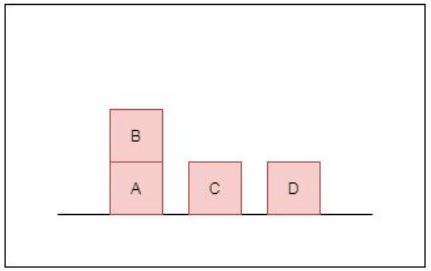


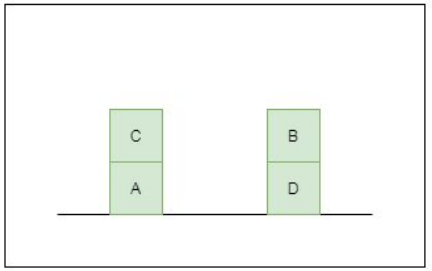

```
(:types block)
(:predicates (on ?b1 ?b2 - block)
             (clear ?b - block)
             (handempty)
             (holding ?b - block))

(:action stack
       :parameters (?b1 ?b2 - block)
       :precondition (and (holding ?b1) (clear ?b2))
       :effect
       (and (not (holding ?b1))
       (not (clear ?b2))
       (clear ?b1)
       (handempty)
       (on ?b1 ?b2)))
```

```
(:init
(handempty)(ontable D)(clear D)(on B A) ...)

(:goal (and
(on B D)(ontable D)
(on C A)(ontable A))))
```





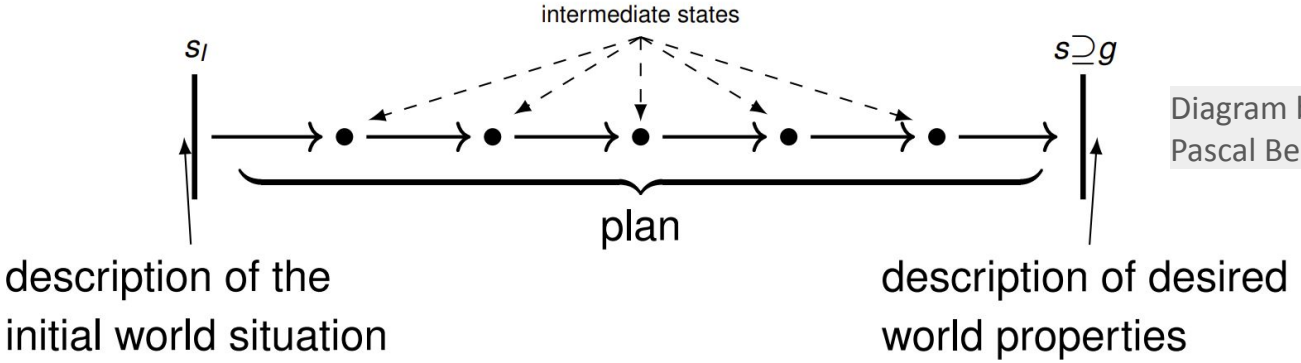
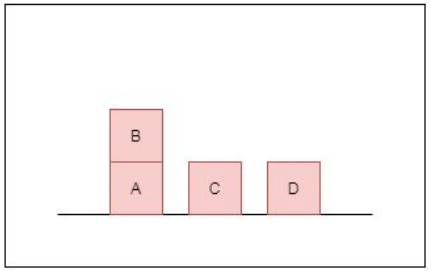


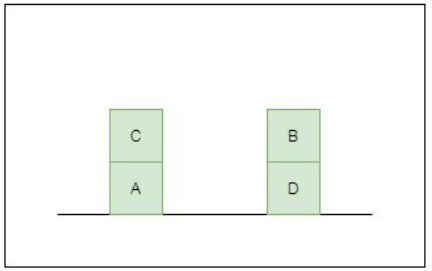
Diagram by
Pascal Bercher

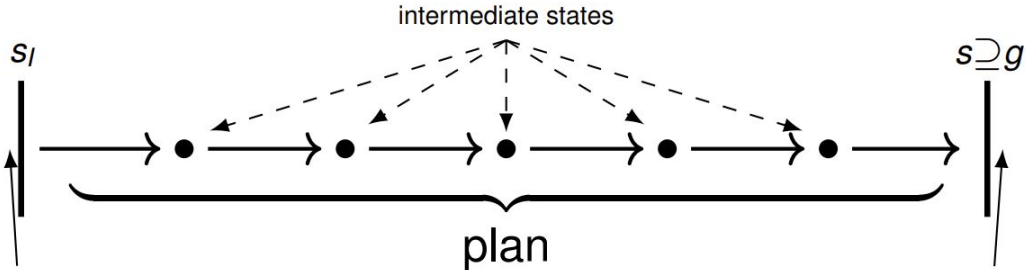
Classical planning: discrete, deterministic, fully observable and single agent.

Problem description induces a state transition system, with nodes as states and actions as transitions.

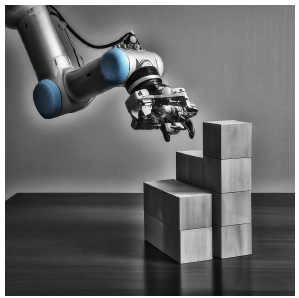
Becomes very complex very quickly: state space is $2^{|V|}$, where V is the number of state features

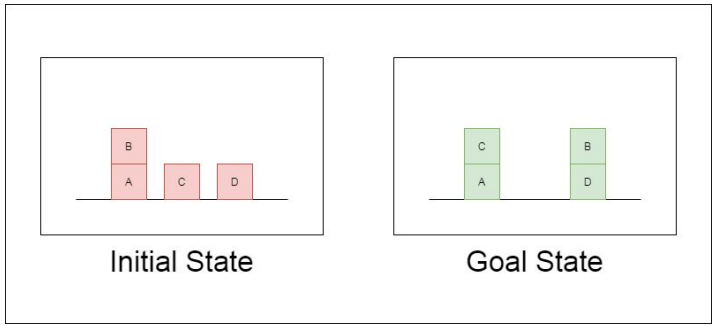


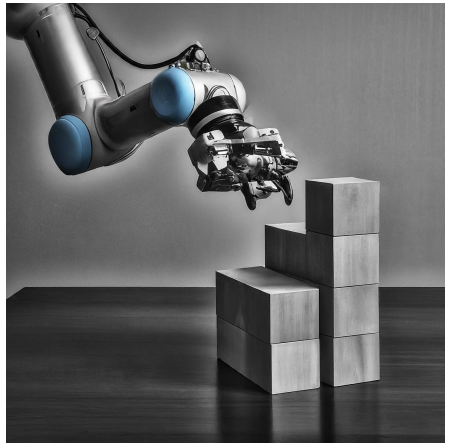




Pickup B -> Stack B on D -> Pickup C -> Stack C on A



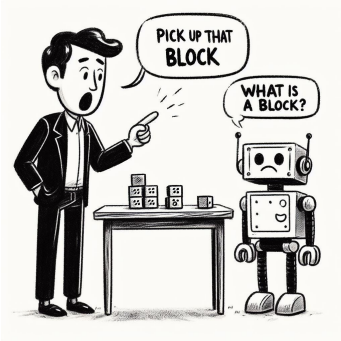




How does it know what a block is ?

How does it know how to stack, pick up and put down?

How does it know what’s the best way to pick up?



Outline

Motivation for integrating task and motion planning

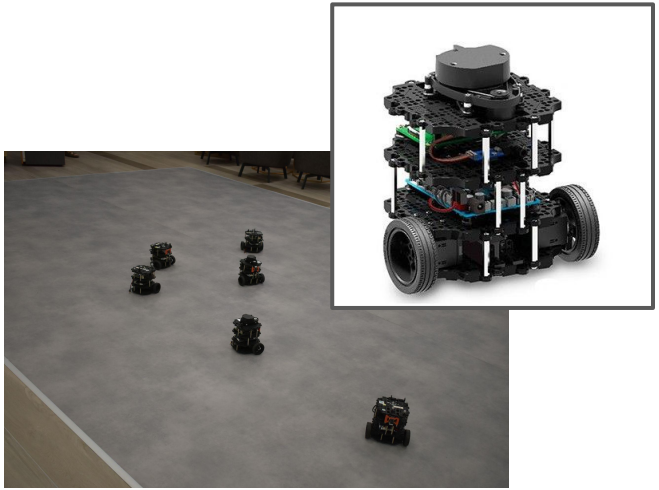
Components of a robotic agent

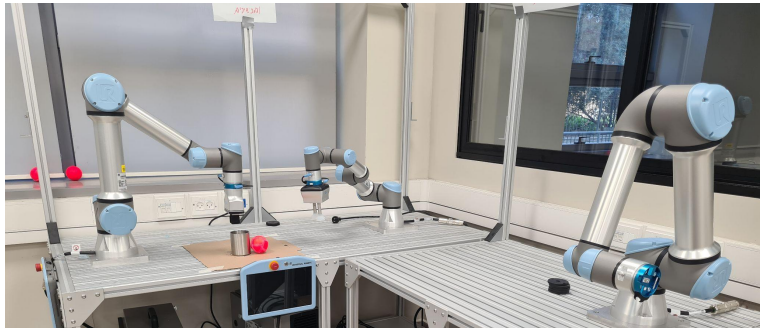
Task planning and its limitations

Motion planning and its limitations

Integrating task and motion planning

Many Types of Robots





Robot Elements

Sensors

Controllers

Actuators

Image from: https://ieeexplore.ieee.org/abstract/document/6386486/

Robotic Agents

state estimation

Maintain a **belief** $\beta \in \boldsymbol{\beta}$ as a probability distribution over world states **S**

$$\beta: S \rightarrow [0,1]$$

decision making

Find a **policy**, mapping **belief** β **and objective into** actions (probabilities)

$$\pi: \boldsymbol{\beta} \square A \rightarrow [0,1]$$

motion control

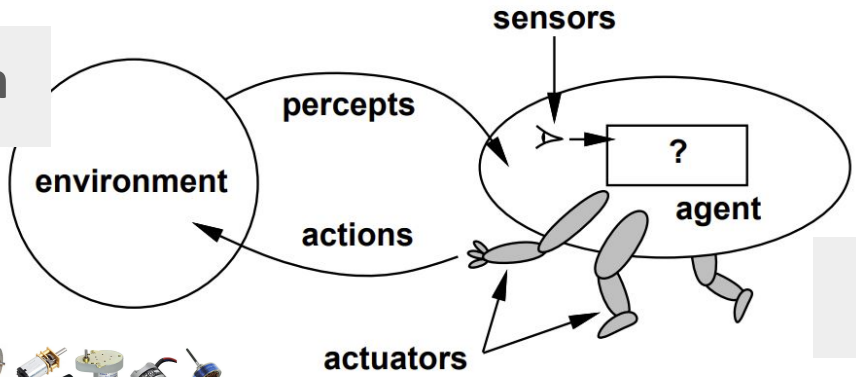
Translate actions into low-level commands (and monitor their execution)

$$u = k_p \cdot e + k_i \cdot \int e dt + k_d \cdot \frac{d}{dt} e$$

https://eng-resources.charlotte.edu/unccengkit/coding/pid-control/

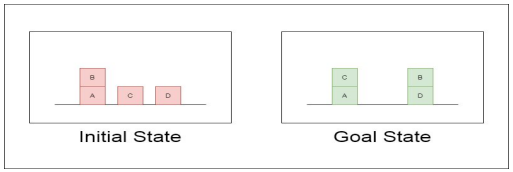
Robotic Agents

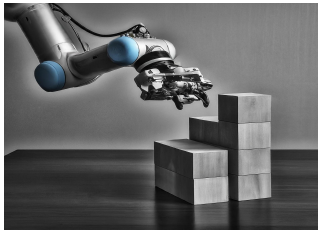
state estimation



decision making

motion control





decision making

```
(:action stack
        :parameters (?b1 ?b2 - block)
        :precondition (and (holding ?b1) (clear ?b2))
        :effect
        (and (not (holding ?b1))
        (not (clear ?b2))
        (clear ?b1)
        (handempty)
        (on ?b1 ?b2)))
```

state estimation

motion control

Task planning is not enough

Things get messier: N-table Blocks World

Accounting for varying costs, collision avoidance, motion feasibility, multiple (infinite) positions from which an action can be performed, multiple (infinite) ways to reach a position

Outline

Motivation for integrating task and motion planning

Components of a robotic agent

Task planning and its limitations

Motion planning and its limitations

Integrating task and motion planning

Task Planning - Classical Formulation

$\langle S, s_0, S_G, A, f, c \rangle$

State space induced by feature set V

- S is a finite and discrete set of states,
- $s_0 \in S$ is the *known initial state*,
- $S_G \subseteq S$ is the non-empty set of goal states,
- $A(s) \subseteq A$ represents the set of actions in A that are applicable in each state $s \in S$,
- $f(a, s)$ is the *deterministic transition function* where $s' = f(a, s)$ is the state that follows s after doing action $a \in A(s)$, and
- $c(a, s)$ is a *positive cost* for doing action a in the state s.

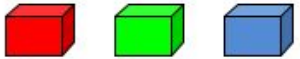
A solution or *plan* in this model is a sequence of applicable actions $a_0, \ldots, a_n$ that generates a state sequence $s_0, s_1, \ldots, s_{n+1}$ where s_{n+1} is a goal state. More precisely, the action a_i is applicable

Geffner, Hector, and Blai Bonet. *A concise introduction to models and methods for automated planning*. Springer Nature, 2022.

Task Planning - Classical Formulation

Action/model:

Objects (things):



Predicates (true or false statements about the world, “facts”):

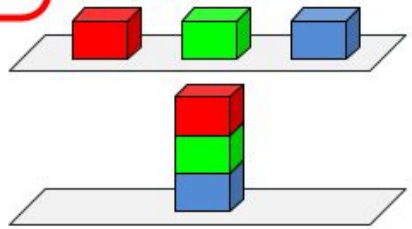
(On A, B)

(Clear A)

Actions (how to change the predicates):

Initial: (Clear) (Clear) (Clear)

Goal: (On ,) (On ,)



slide by Erez Karpas

Task Planning: Solution approaches

Common algorithms/techniques:

- Search – forward, regression, bidirectional, symbolic
- Compilation – to SAT, to CP
- Partial Order Planning
- Dynamic programming: Bellman Backup – i.e. for problems with uncertainty and rewards, like MDPs and POMDPs.

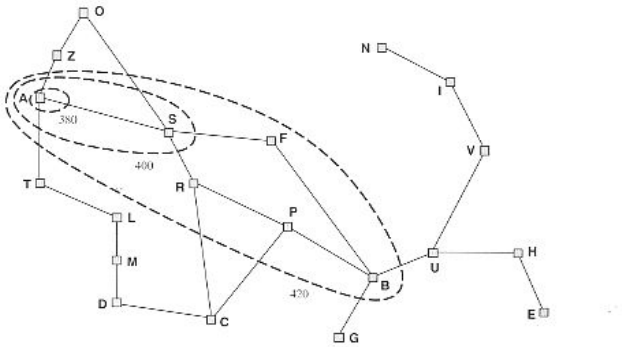
What are some general search algorithms?

Best First Search

```
function BEST-FIRST-SEARCH(problem, f) returns a solution node or failure
  node ← NODE(STATE=problem.INITIAL)
  frontier ← a priority queue ordered by f, with node as an element
  reached ← a lookup table, with one entry with key problem.INITIAL and value node
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
      s ← child.STATE
      if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
        reached[s] ← child
        add child to frontier
  return failure

function EXPAND(problem, node) yields nodes
  s ← node.STATE
  for each action in problem.ACTIONS(s) do
    s' ← problem.RESULT(s, action)
    cost ← node.PATH-COST + problem.ACTION-COST(s, action, s')
    yield NODE(STATE=s', PARENT=node, ACTION=action, PATH-COST=cost)
```

In our implementation, we check if the node is terminal



Code above from Norvig, P. Russel, and S. Artificial Intelligence. "A modern approach." *Prentice Hall Upper*

Pearl, Judea. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., 1984.

What is not captured by classical task planning ?

Task Planning - Extensions

Accounting for stochastic action outcome using a state transition probability function

$$\mathcal{P}^a_{s,s'} = \mathcal{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

Accounting for objectives using a reward function

$$\mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$

Accounting for reward deprecation using a discount factor

$$\gamma \in [0, 1]$$

Accounting for partial observability using observations Ω and observation function

$$\mathcal{O}^o_{s,a} = \mathcal{P}[O_{t+1} = o | S_t = s, A_t = a]$$

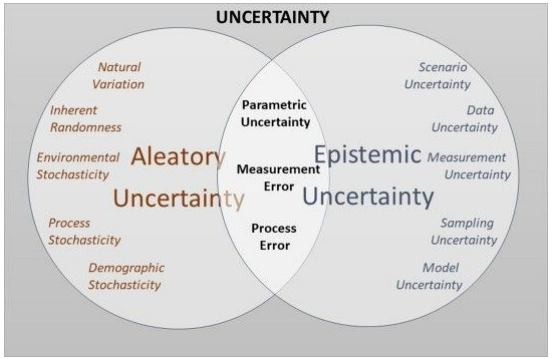
Accounting for action duration, multiple agents, varying costs, etc.

Sequential decision making: what should the agent do to maximize total rewards?

What is not captured by task planning ?

Task Planning vs. the Real World

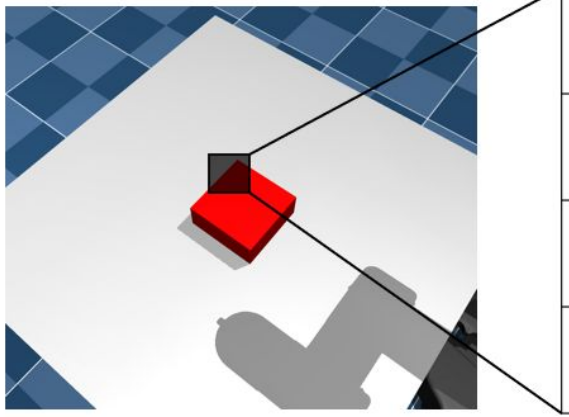
- **Discrete:** plans are finite sequences of actions - while actual actions are continuous
- **Fully modeled actions** (deterministic in classical planning): while actual actions can have unexpected effects that are hard/impossible to model
- **Full knowledge** of world state or of belief (distribution over possible world states) while knowing the actual state /distribution relies on complex sensing and reasoning



https://www.stat.berkeley.edu/users/aldous/157/Papers/Fox_Ulkumen.pdf

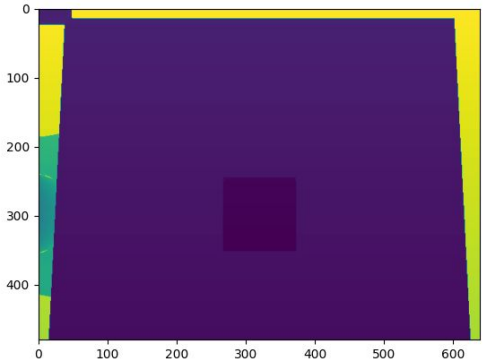
State Estimation

How do we know that block B is on Table 3?



(128, 128, 156)	(128, 62, 53)	(96, 128, 159)	(254, 128, 156)
(122, 124, 200)	(111, 138, 156)	(128, 128, 19)	(252, 128, 156)
(122, 198, 96)	(155, 128, 156)	(242, 128, 156)	(201, 113, 157)
(128, 18, 102)	(255, 181, 156)	(199, 178, 156)	(129, 176, 196)

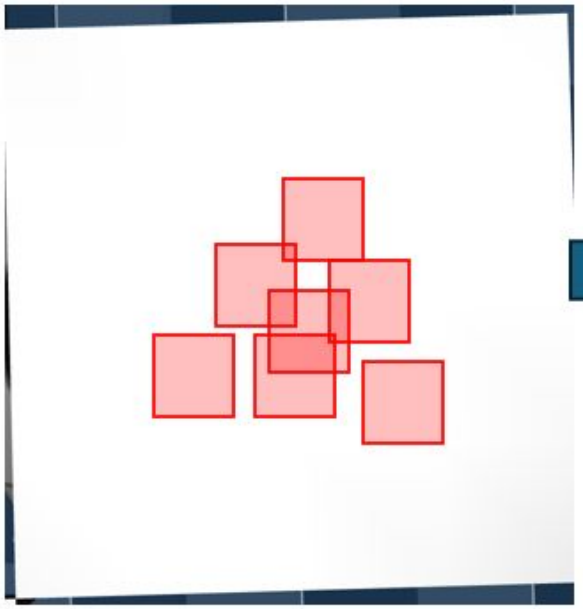
RGB image



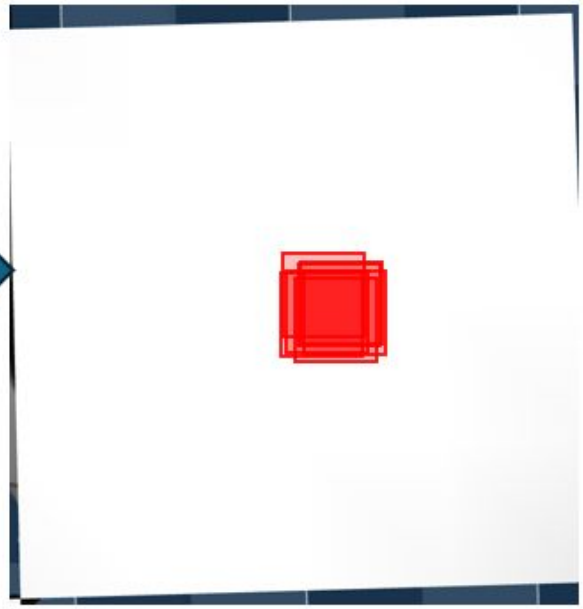
Depth image

Belief Update

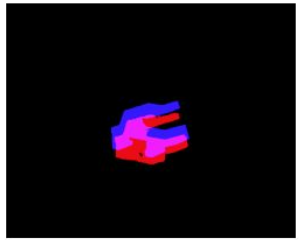
β

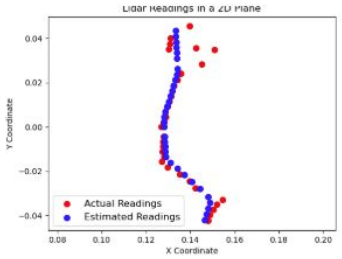


β'



Belief Update





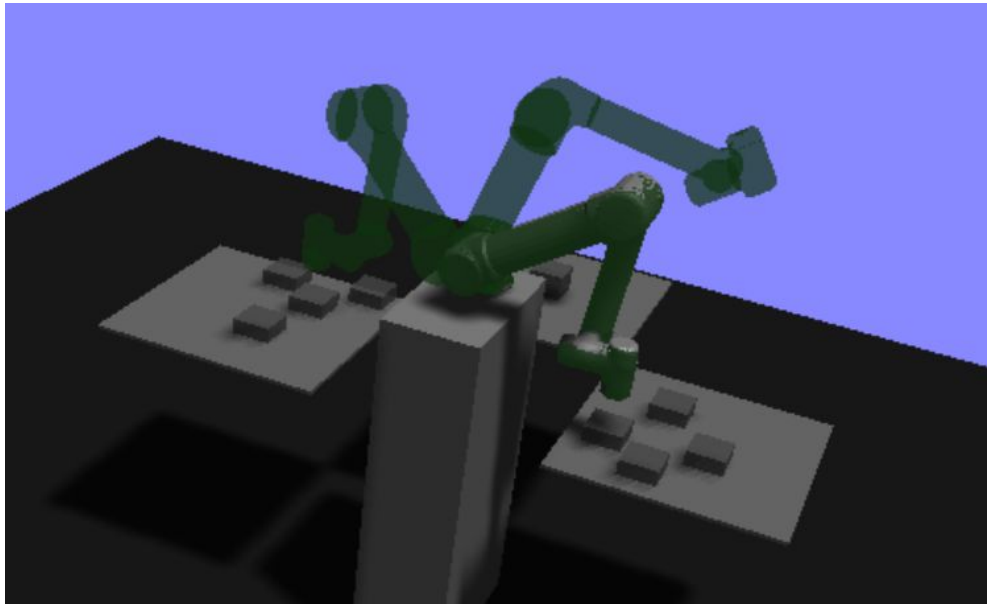
(a) P1 (b) P2 (c) P3 (d) P4 (e) P5 (f) P6

Motion Planning

How to attempt to pick up block B ? How to reach that position ?

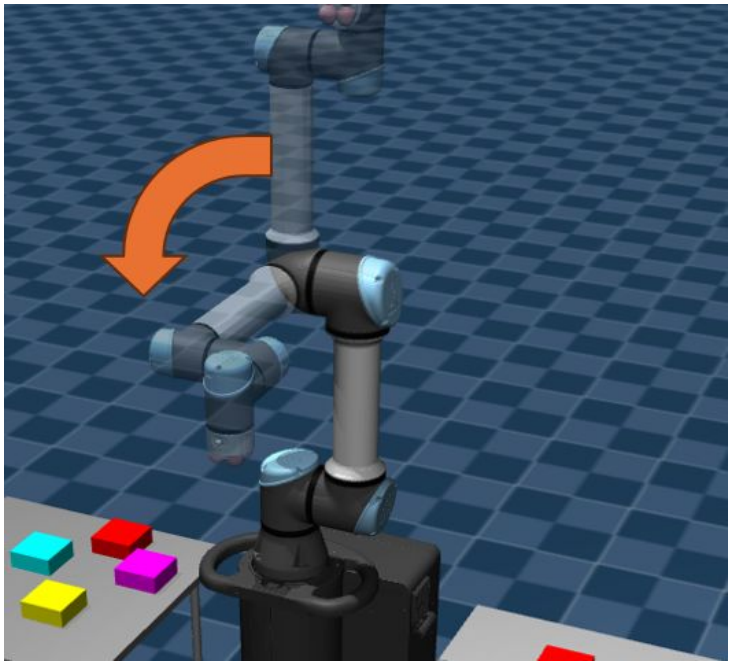
Motion Planning

How to get from one position to another ?



Control and Monitoring

how much force to apply?



move fast vs. accurate the motions should be ?

how to verify that the block was actually picked up (and not dropped)?

Outline

Motivation for integrating task and motion planning

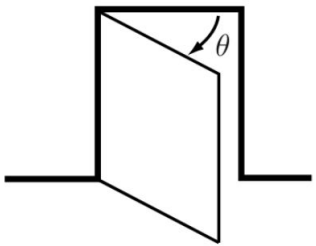
Components of a robotic agent

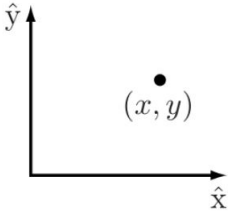
Task planning and its limitations

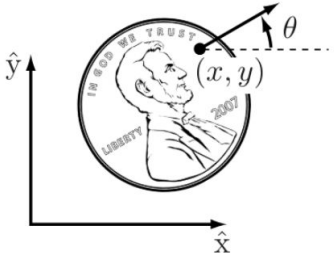
Motion planning and its limitations

Integrating task and motion planning

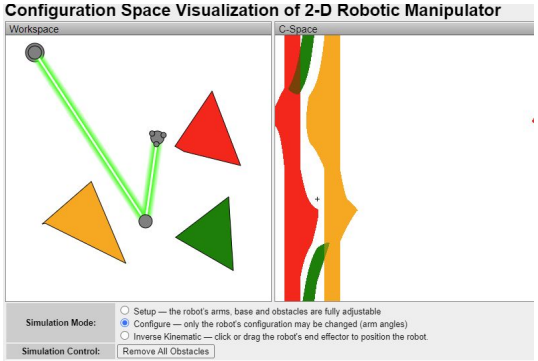
Configuration Space and Degrees of Freedom (DOF)



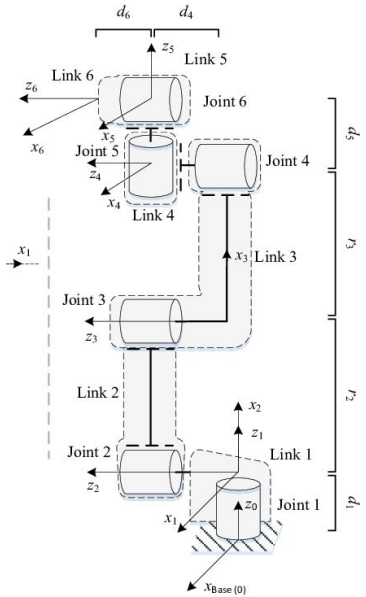




https://hades.mech.northwestern.edu/images/7/7f/MR.pdf

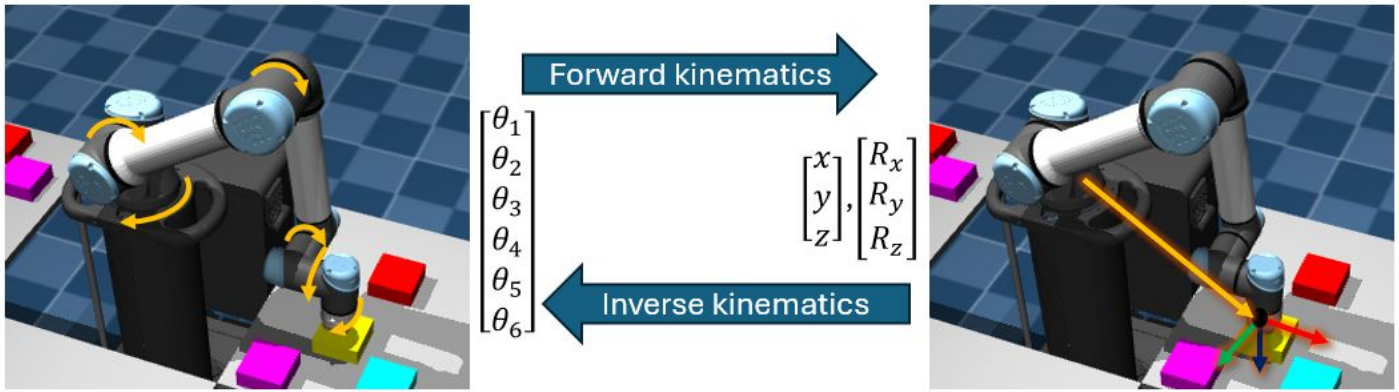


https://www.cs.unc.edu/~jeffi/c-space/robot.xhtml



Galin, Rinat, Roman Meshcheryakov, and Anna Samoshina. "Mathematical modelling and simulation of human-robot collaboration." *2020 International Russian Automation Conference (RusAutoCon)*. IEEE, 2020.

Configuration Space

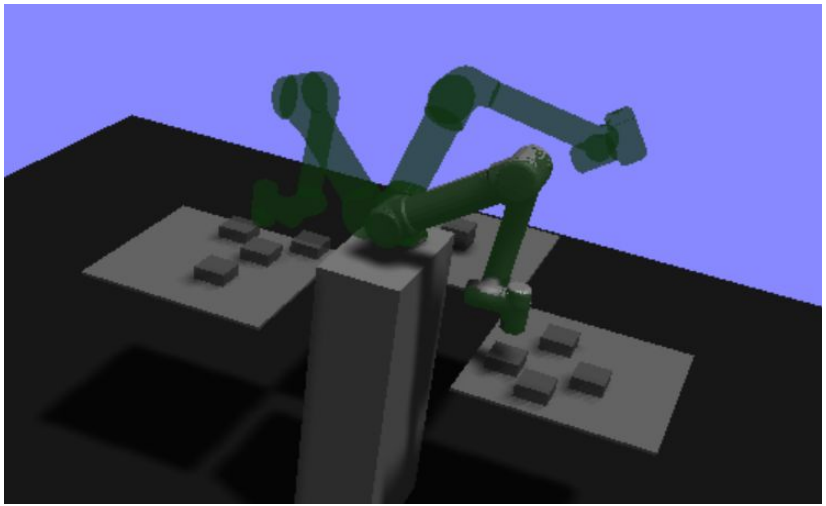


Translating between joint states and end effector pose

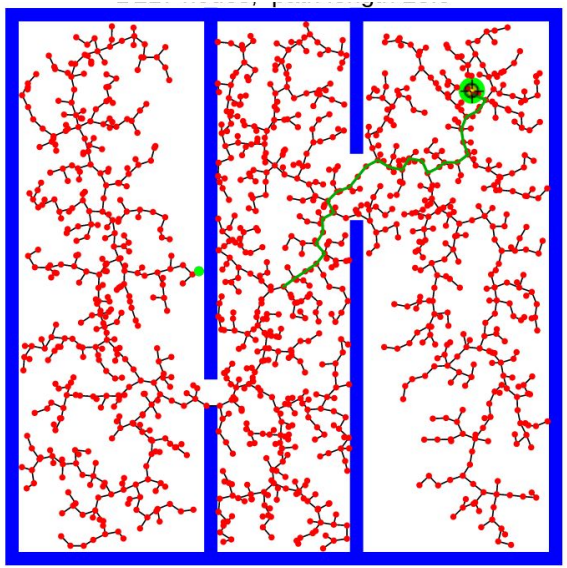
Motion Planning Problem in configuration space (C-Space)

Given configurations x_i, $x_g \in X$, compute a path from x_i to x_g

Path: continuous function τ:[0,1] $\longrightarrow$ X s.t. $\tau(0) = x_i$ and $\tau(1)=x_g$ and $\forall x\ \tau(1)$ not in X_{obs}

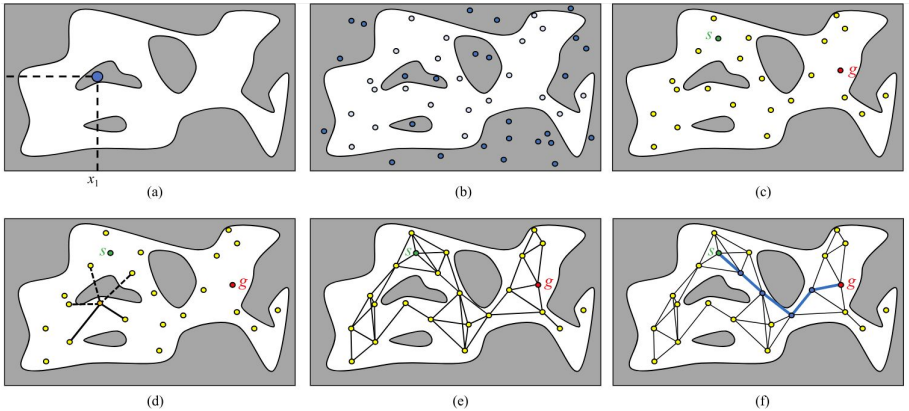


Many approaches (we will see some in the labs)



Rapidly Exploring Random Trees

https://demonstrations.wolfram.com/RapidlyExploringRandomTreeRRTAndRRT/



Probabilistic Road Map (PRM)

http://motion.cs.illinois.edu/RoboticSystems/figures/planning/prm.svg

What is not captured by motion planning ?

Task planning vs. motion planning

Task planners can reason over very large sets of states by manipulating partial descriptions, while geometric **motion planners** operate on completely detailed specifications of world states.

A task planner could decide that the living room needs to be traversed, disregarding the detailed arrangement of its furniture.

Motion planners deal beautifully with geometry, but not with non-physical aspects of the domain; they can plan how to get to the phone but not decide that a phone call needs to be made.

Explanation as I heard it from Leslie Kaelbling

Outline

Motivation for integrating task and motion planning

Components of a robotic agent

Task planning and its limitations

Motion planning and its limitations

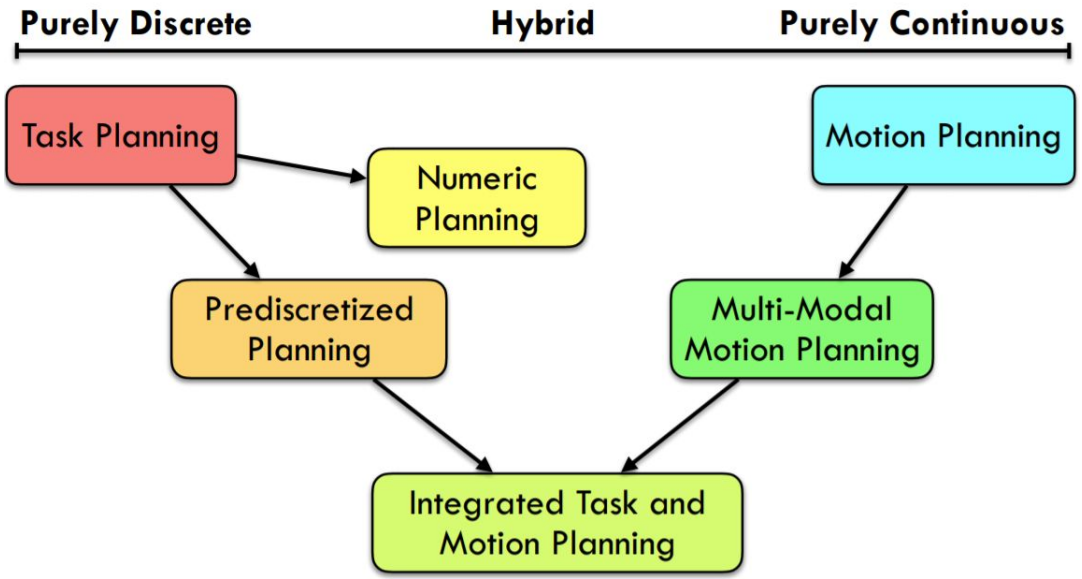
Integrating task and motion planning

Task and Motion Planning: Challenges

- Inherits challenges of both motion & task planning
  - high-dimensional, continuous state-spaces
  - long horizons

- Geometric and continuous constraints limit high-level strategies
  - kinematics, reachability, joint limits, collisions, grasp, visibility, stability, stiffness, torque limits,

Approaches to Task and Motion Planning



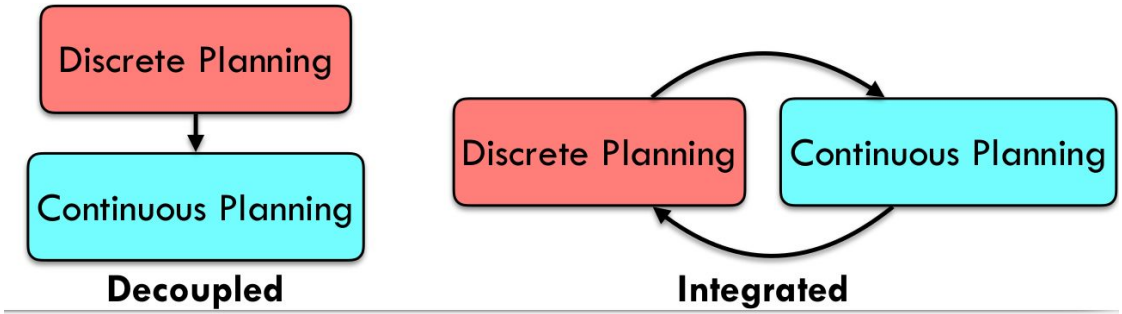
Decoupled vs. Integrated Task and Motion Planning

Decoupled:

- discrete (task) planning then continuous (motion) planning
- requires a strong **downward refinement assumption:** every correct discrete plan can be refined into a correct continuous plan (hierarchical planning)

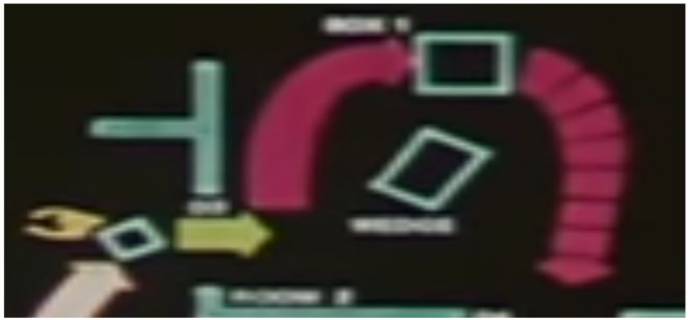
Integrated:

- simultaneous discrete & continuous planning



Shakey the Robot (1969): Task planning then motion planning

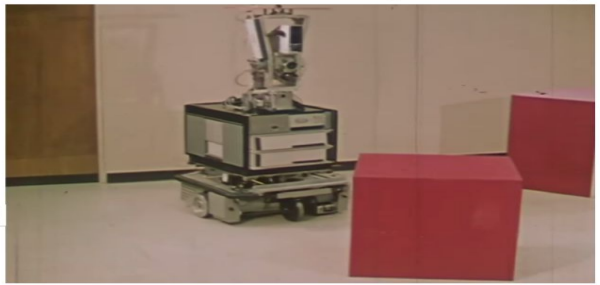
- First autonomous mobile manipulator (via pushing)
- Visibility graph, A* search, and STRIPS (which is a grounded form of PDDL) [Fikes 1971, Nilson 1984]



```
type(robot robot)      type(o1 object)
name(robot shakey)     name(o1 box1)
at(robot 4.1 7.2)      at(o1 3.1 5.2)
theta(robot 90.1)      inroom(o1 r1)
                       shape(o1 wedge)
                       radius(o1 3.1)
```

GOTHRU(d,r1,r2)

Precondition	INROOM(ROBOT,r1) ∧ CONNECTS(d,r1,r2)
Delete List	INROOM(ROBOT,$)
Add List	INROOM(ROBOT,r2)



Can we suggest better ways ?