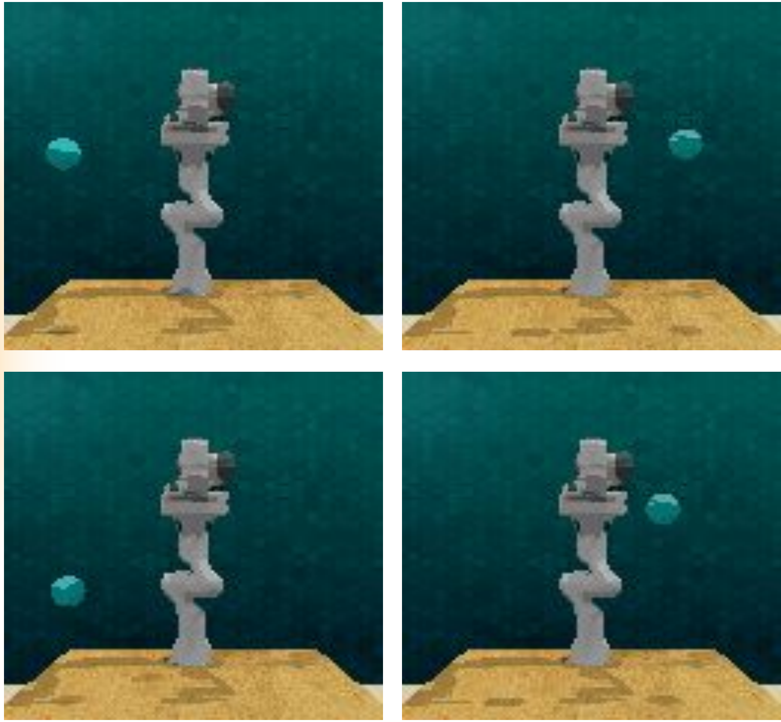
DeepRob

[Student] Lecture 22

by Chahyon Ku, Carl Winge, Aaron Fernandes

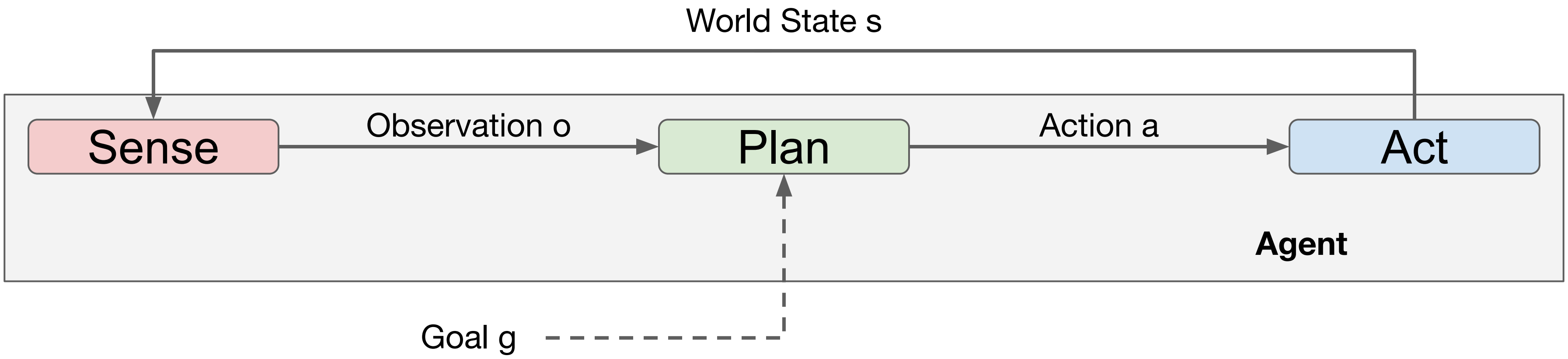
Imitation Learning, Visual-Language Models for Robot Manipulation

University of Michigan and University of Minnesota

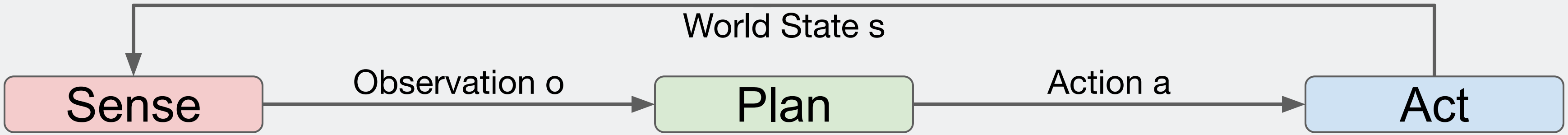


Sense Plan Act

Sense Plan Act



Preliminaries



Partially Observable Markov Decision Process (**POMDP**)

- A Markov decision process is a framework for decision making in an environment where outcomes are partly random, and partly determined by the agent's actions
- The state is not fully observable, so the agent relies on sensor observations, like a camera view

Sense

World State s

Sense → Observation o → Plan → Action a → Act

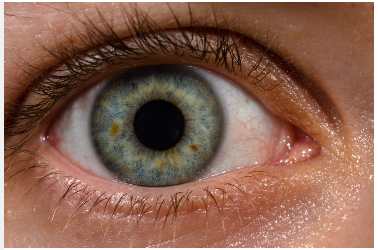
External (Environmental) Observations

- Eyes (Camera, LIDAR, RADAR)
- Ears (Microphone)
- Nose

Internal (Proprioceptive) Observations

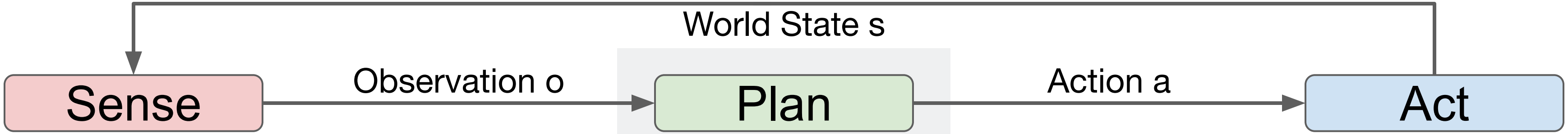
- Muscles (Joint Angles, Velocities, Torques)
- Inertial Measurement Units (IMU)

Wikipedia - Human Eye



Intel - L515 Lidar Camera

Plan



Policy: Function **π** mapping observation **o** to action **a** to maximize **return.**

Plan

World State s

Sense → Observation o → Plan → Action a → Act

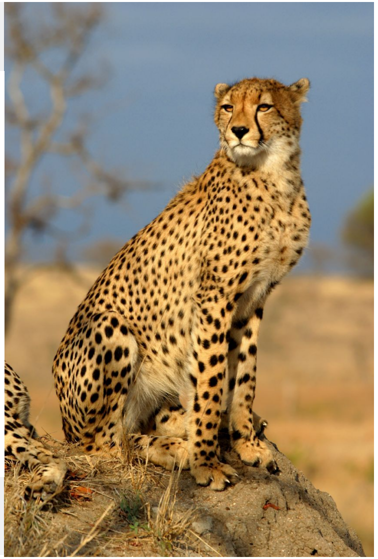
Policy: Function **π** mapping observation **o** to action **a** to maximize **return.**

What Return (Goal)?

- For natural agents? **Survive and reproduce**
- For robotic agents? **Serve humans**

How can we create policies that serve humans best?

Fetch Robotics
Fetch Mobile Manipulator



Wikipedia - Cheetah

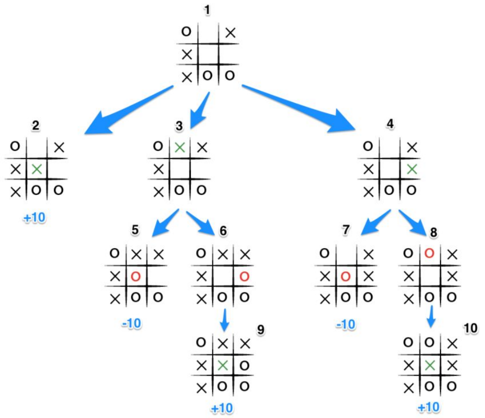
Plan

World State s

Sense → Observation o → Plan → Action a → Act

How can we create such policies?

- Algorithmic (Rule-based, **Search**, Sampling, etc.)



never stop building
minimax

Wikipedia
A* Search Algorithm

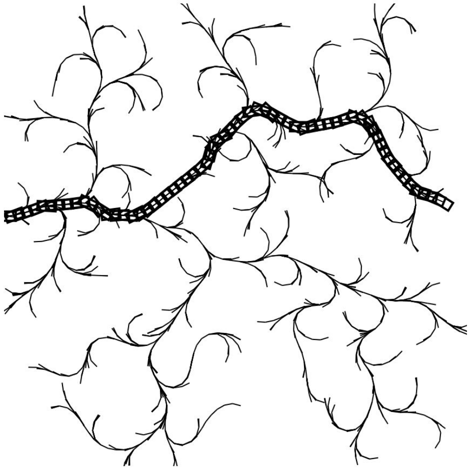
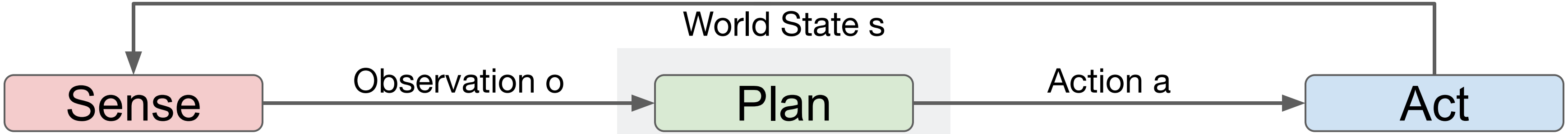


Figure 1: A 2D projection of a 5D RRT for a kinodynamic car.

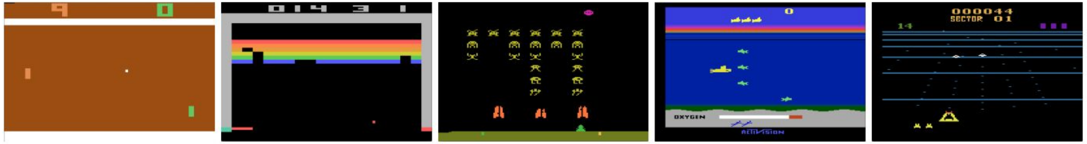
Rapidly-Exploring Random Trees: A New Tool For Path Planning (LaValle 1998)

Plan



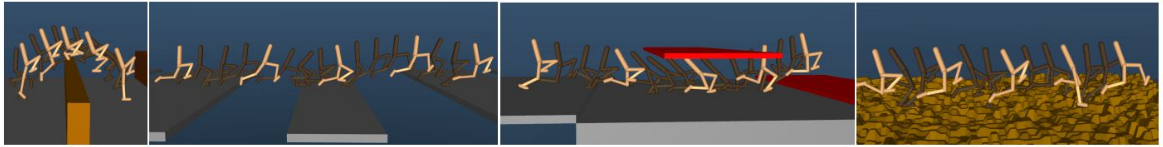
How can we create high-dimensional policies? Learning!

- Reinforcement Learning (Learning from Exploration)
  - "**Try a lot** of things. Figure out what works **best**"



Playing Atari with Deep Reinforcement Learning (NeurIPS 2014)

(Deepmind, arxiv 2017)
Emergence of Locomotion Behaviours in Rich Environments



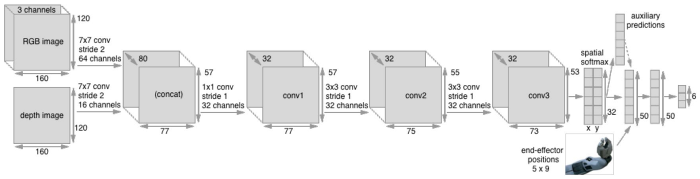
Plan

World State s

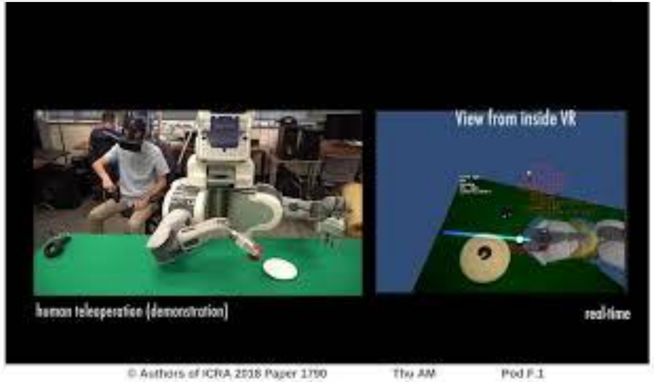
Sense → Observation o → Plan → Action a → Act

How can humans create high-dimensional, data-efficient policies?

- Imitation Learning (Learning from Demonstration)
  - “Get a (human) **expert to demonstrate** the task. **Copy** it as best as you can”

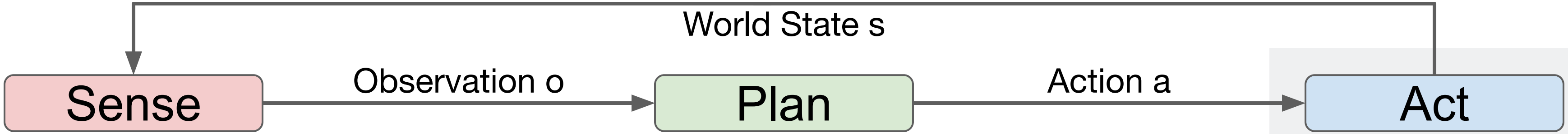


task	reaching	grasping	pushing	plane	cube	nail	grasp-and-place	grasp-drop-push	grasp-place-x2	cloth
test	91.6%	97.2%	98.9%	87.5%	85.7%	87.5%	96.0%	83.3%	80%	97.4%
demo time (min)	13.7	11.1	16.9	25.0	12.7	13.6	12.3	14.5	11.6	10.1
avg length (at 10 Hz)	41	37	58	47	37	38	68	87	116	60
# demo	200	180	175	319	206	215	109	100	60	100



Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation (Zhang et al. ICRA 2018)

Act



Discrete

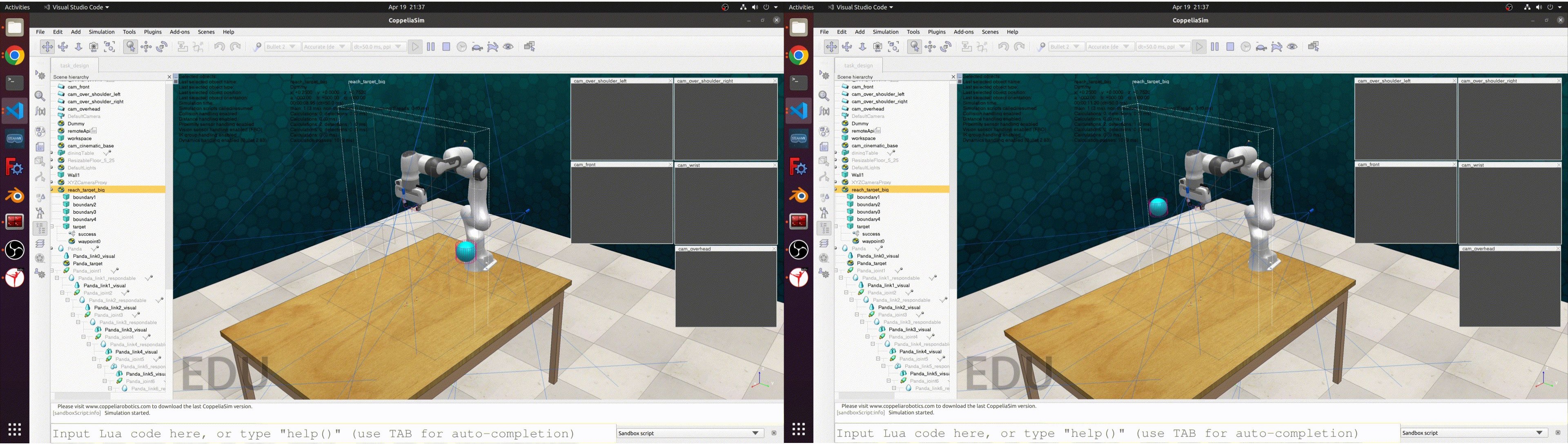
- Up, Down, Left, Right (Atari, GridWorld)
- Gripper open or close all the way

Continuous

- Acceleration (Vehicles)
- Steering (Vehicles)
- Joint Position, Velocity, Torque (Robot Arm)
- End-effector Pose or Velocity (Robot Arm)
- Gripper position (Robot Arm)

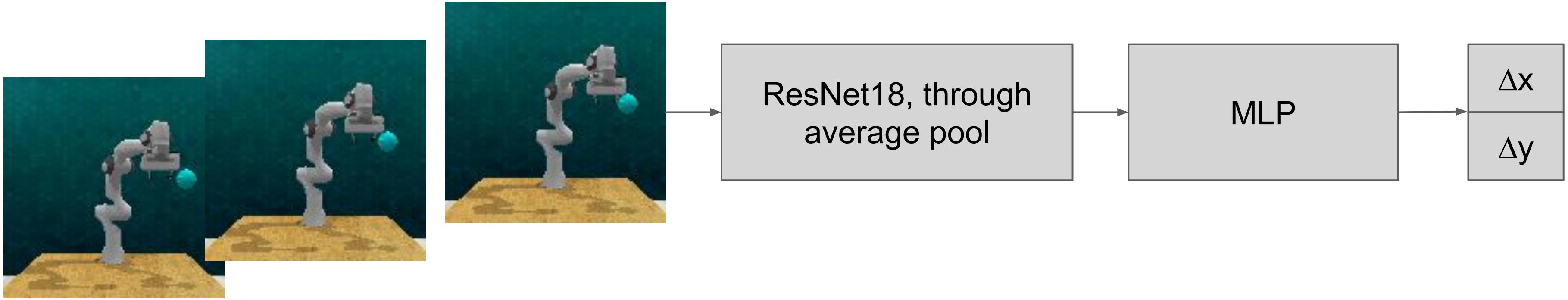
Simple Imitation Learning Example

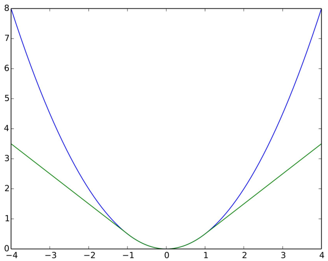
Collect Demonstrations



Examples of procedural demonstrations in CoppeliaSim using RLBench. The robot moves to the cyan target, which is randomly generated in plane with the robot end effector.

Train the Network





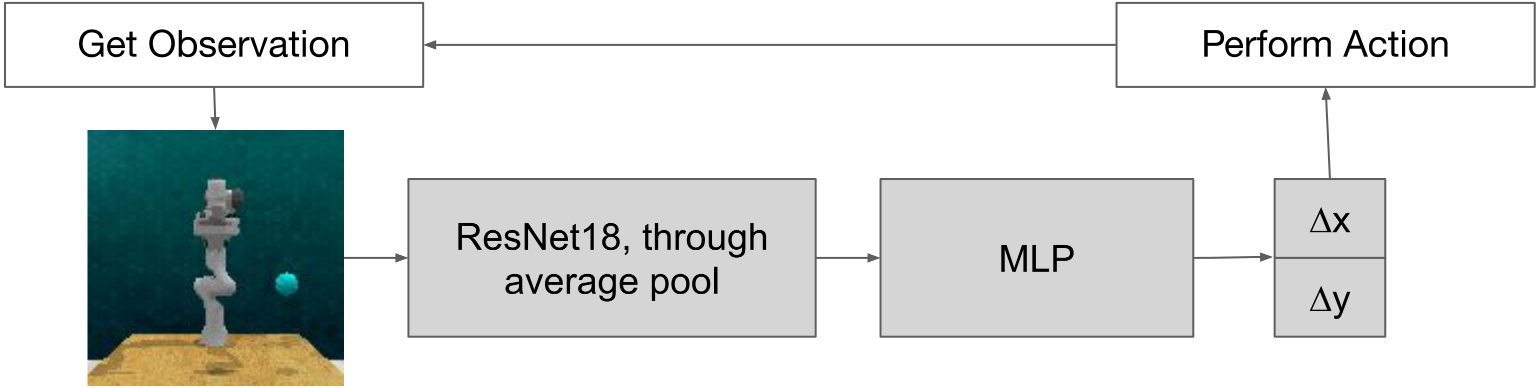
Wikipedia

Huber Loss (PyTorch)

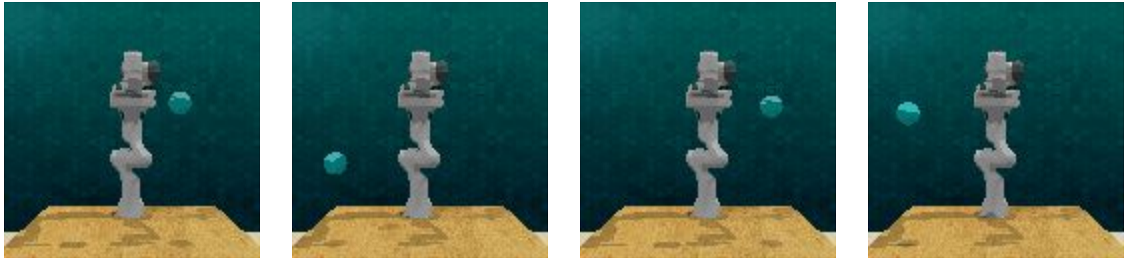
$$l_n = \begin{cases} 0.5(x_n - y_n)^2, & \text{if } |x_n - y_n| < delta \\ delta * (|x_n - y_n| - 0.5 * delta), & \text{otherwise} \end{cases}$$

where x_n are predictions and y_n are ground truth values

Evaluate



Results



Summary

- Sense Plan Act
  - Agent
  - POMDP
- Policy
  - Algorithmic
  - Reinforcement Learning (Learning from Exploration)
  - Imitation Learning (Learning from Demonstration)

- What about higher dimensional action space and longer horizon goals?

Long Horizon Planning

Goals of Robots

- Using the control policies from the previous section robots can now perform simple tasks
- We want robots to perform complex tasks
- How do we perform complex tasks?

Control vs. Planning

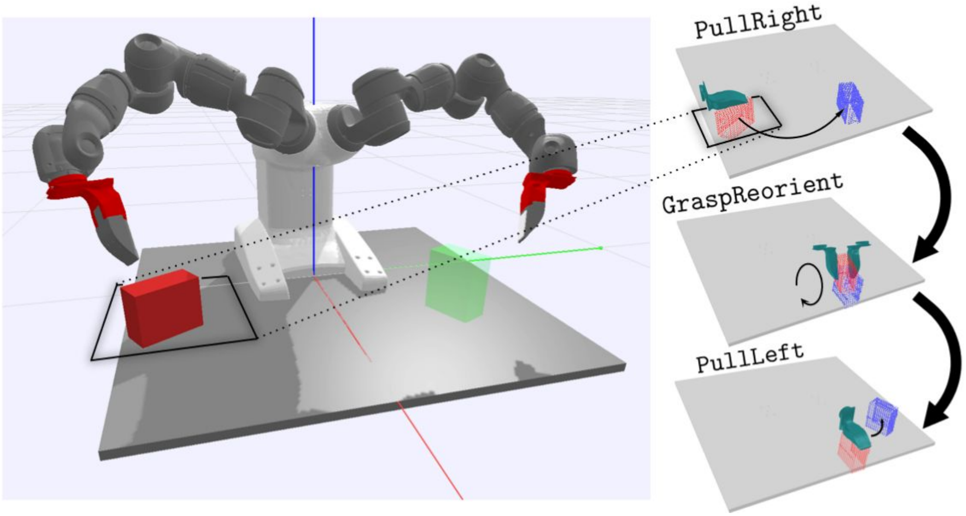
- Control is low level: motor torque, velocity, and position
- Motion planning uses motor controls to complete a trajectory
- Task planning typically requires several trajectories in sequence to achieve a goal state
- Several sub-tasks may be needed to complete a complex task

Short Horizon vs. Long Horizon

- Short horizon task examples push, pull, place, etc.
- Long horizon task examples clean up spill, fetch food
- Control policies learn short horizon tasks
  - Cannot do long horizon tasks

Long Horizon Tasks

- Large search space of low level tasks
- Shrink the search space by creating sub goals



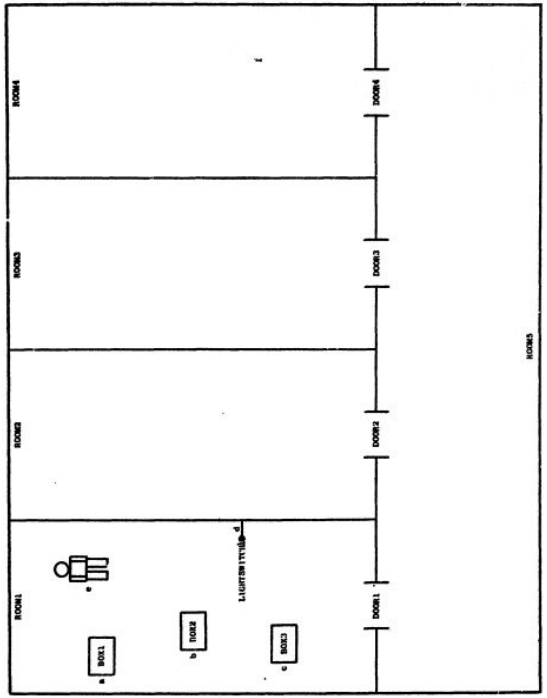
A Long Horizon Planning Framework for Manipulating Rigid Pointcloud Objects (CoRL 2020)

Breaking Long Horizon Tasks into Subgoals

- Symbolic Planning
  - Finds effective subgoals
  - Needs explicit primitives and constraints
  - Uses predicate calculus to represent the world and actions
  - Uses the possible actions and world state to formulate plans

STRIPS: A New Approach to the Application of .Theorem Proving to Problem Solving
(IJCAI 1971)

Breaking Long Horizon Tasks into Subgoals



$$M_0: \left\{ \begin{array}{l} \mathrm{ATR}(a) \\ \mathrm{AT}(\mathrm{BOX1}, b) \\ \mathrm{AT}(\mathrm{BOX2}, c) \\ \mathrm{AT}(\mathrm{BOX3}, d) \end{array} \right\} .$$

World Model

The goal wff describing this task is

$$G_0: (\exists x)\,[\mathrm{AT}(\mathrm{BOX1}, x) \wedge \mathrm{AT}(\mathrm{BOX2}, x) \wedge \mathrm{AT}(\mathrm{BOX3}, x)].$$

Goal

(1) *push* (k, m, n): Robot pushes object k from place m to place n.

Precondition: $\mathrm{AT}(k, m) \wedge \mathrm{ATR}(m)$

Negated precondition: $\sim \mathrm{AT}(k, m) \vee \sim \mathrm{ATR}(m)$

Delete list: $\mathrm{ATR}(m)$, $\mathrm{AT}(k, m)$

Add list: $\mathrm{AT}(k, n)$, $\mathrm{ATR}(n)$

(2) *goto*(m, n): Robot goes from place m to place n.

Precondition: $\mathrm{ATR}(m)$

Negated precondition: $\sim \mathrm{ATR}(m)$

Delete list: $\mathrm{ATR}(m)$

Add list: $\mathrm{ATR}(n)$.

Operators

STRIPS: A New Approach to the Application of .Theorem Proving to Problem Solving (IJCAI 1971)

Semantic Planning

-The world model describes the initial state

-In this example the initial state is that the robot is at location a, box1 is at location b, box2 is at location c, and box 3 is at location d

$$M_0: \begin{Bmatrix} \mathrm{ATR}(a) \\ \mathrm{AT}(\mathrm{BOX1}, b) \\ \mathrm{AT}(\mathrm{BOX2}, c) \\ \mathrm{AT}(\mathrm{BOX3}, d) \end{Bmatrix}$$

STRIPS: A New Approach to the Application of .Theorem Proving to Problem Solving (IJCAI 1971)

Semantic Planning

- Goal needs to be stated in a way which allows a logical plan
- STRIPS calls this a well formulated formula
- This statement says there exists an x such that box1 is at x and box 2 is at x and box 3 is at x

$$G_0: (\exists x)\,[\mathrm{AT}(\mathrm{BOX1}, x) \wedge \mathrm{AT}(\mathrm{BOX2}, x) \wedge \mathrm{AT}(\mathrm{BOX3}, x)].$$

STRIPS: A New Approach to the Application of .Theorem Proving to Problem Solving (IJCAI 1971)

Semantic Planning

- Operators describe actions the agent can take
- Preconditions must be met that enable an action which in turn updates the world state
- for the example push the precondition is that both the robot and object are at the location m

(1) *push* (k, m, n): Robot pushes object k from place m to place n.

Precondition: $\mathrm{AT}(k, m) \wedge \mathrm{ATR}(m)$

Negated precondition: $\sim \mathrm{AT}(k, m) \vee \sim \mathrm{ATR}(m)$

Delete list: $\mathrm{ATR}(m)$
$\mathrm{AT}(k, m)$

Add list: $\mathrm{AT}(k, n)$
$\mathrm{ATR}(n)$

(2) *goto*(m, n): Robot goes from place m to place n.

Precondition: $\mathrm{ATR}(m)$

Negated precondition: $\sim \mathrm{ATR}(m)$

Delete list: $\mathrm{ATR}(m)$

Add list: $\mathrm{ATR}(n)$.

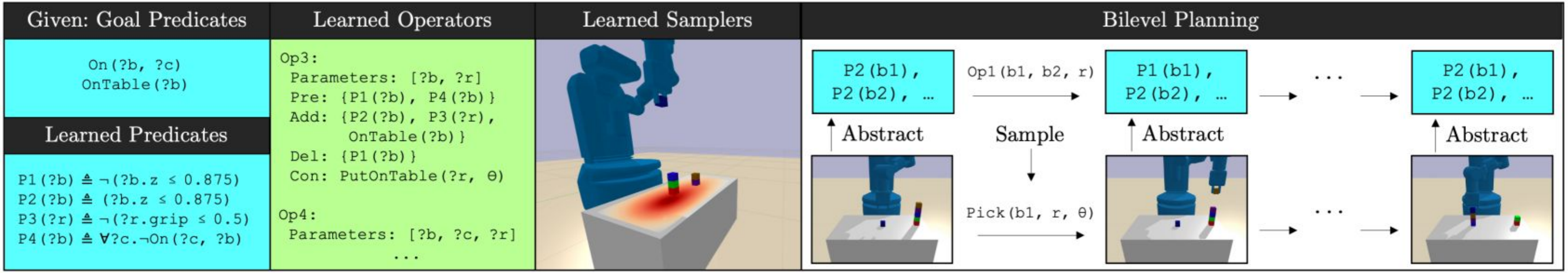
STRIPS: A New Approach to the Application of .Theorem Proving to Problem Solving (IJCAI 1971)

Machine Learning Approaches

- Define these primitives from learning
- Graph Neural Network Planner
- Hierarchical Reinforcement Learning
- Large Language Models

Learning Predicates

- Takes goal predicates and iteratively learns new predicates, operators, and samplers
- Adds intuitive predicates until a feasible plan is given



Predicate Invention for Bilevel Planning (AAAI 2023)

Operator Learning

1. Each demonstration is a set of states and actions which are combined and called dataset transitions
2. The condition, added effect, and deleted effect are created by applying a parameter to each transition
3. From the condition, and effects the precondition can be extrapolated which gives the full operator

Learned Operators

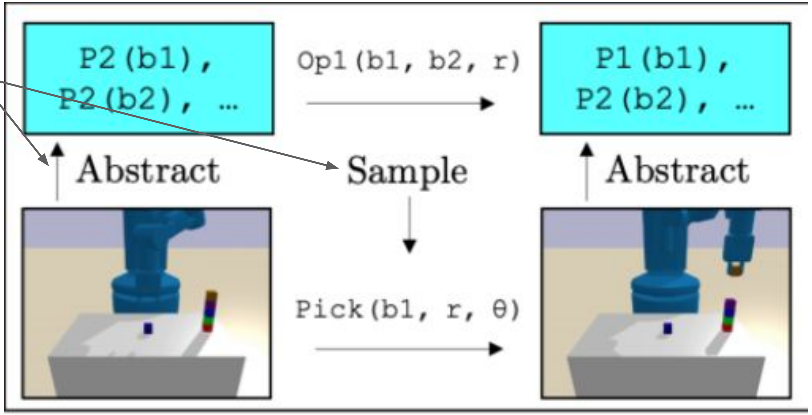
```
Op3:
 Parameters: [?b, ?r]
 Pre: {P1(?b), P4(?b)}
 Add: {P2(?b), P3(?r),
           OnTable(?b)}
 Del: {P1(?b)}
 Con: PutOnTable(?r, θ)

Op4:
 Parameters: [?b, ?c, ?r]
            ...
```

Bilevel Planning

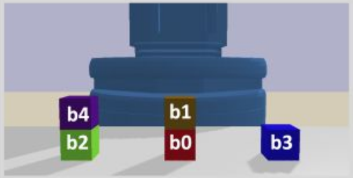
```
PLAN(x_0, g, Ψ, Ω, Σ)
  // Parameters: n_abstract, n_samples.
  s_0 ← ABSTRACT(x_0, Ψ)
  for π̂ in GENABSTRACTPLAN(s_0, g, Ω, n_abstract)
    if π ~ REFINE(π̂, x_0, Ψ, Σ, n_samples) then
      return π
```

Algorithm 1: Pseudocode for our bilevel planning algorithm. The inputs are an initial state x_0, goal g, predicates Ψ, operators Ω, and samplers Σ; the output is a plan π. An outer loop runs GENABSTRACTPLAN, which generates plans in the abstract state and action spaces. An inner loop runs REFINE, which attempts to refine each abstract plan $\hat{\pi}$ into a plan π. If REFINE succeeds, then the found plan π is returned as the solution; if REFINE fails, then GENABSTRACTPLAN continues.



Bilevel Planning

Initial state:



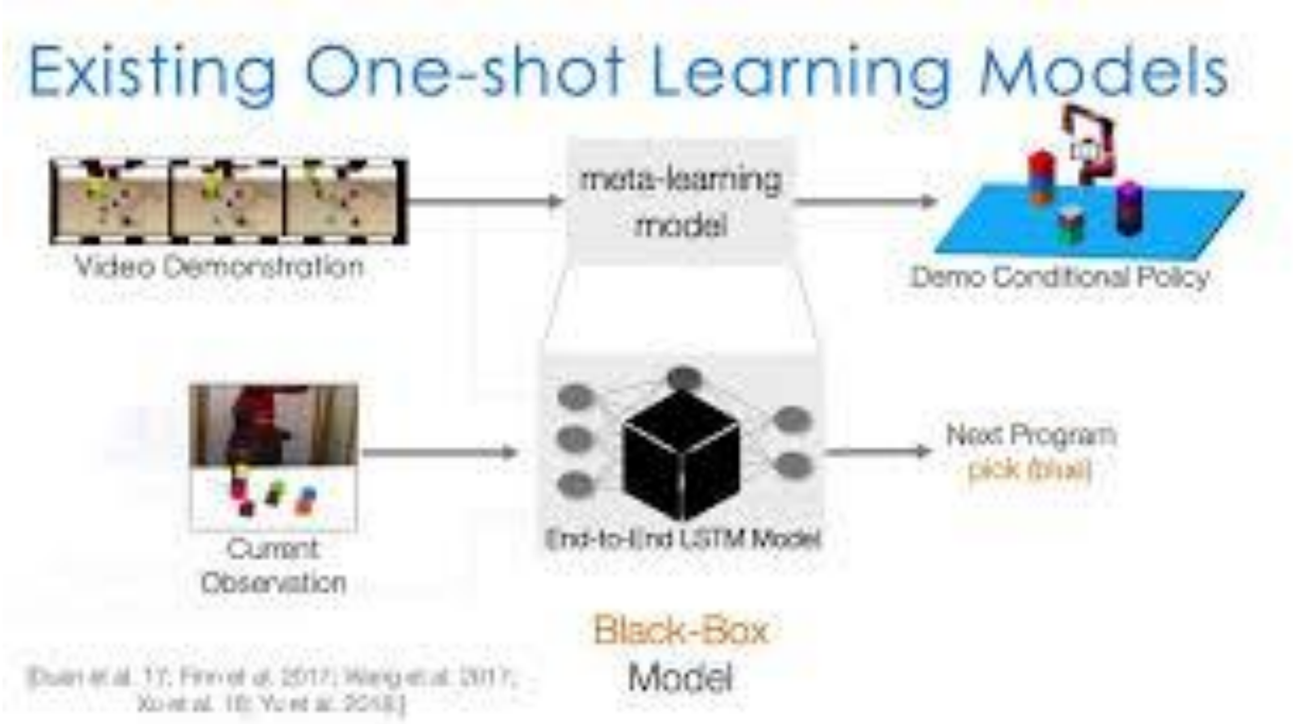
Goal:

`OnTable(b3)`
`On(b2, b3)`
`On(b1, b2)`

Search Iteration	Predicate Set	J_{surr} (lower is better)	Success Rate on Evaluation Tasks
0	On(?b, ?c) OnTable(?b)	$1.3 \cdot 10^7$	0%
1	On(?b, ?c) OnTable(?b) ¬(?b.z ≤ 0.875)	$1.0 \cdot 10^7$	12%
2	On(?b, ?c) OnTable(?b) ¬(?b.z ≤ 0.875) ∀?c.¬On(?c, ?b)	$2.0 \cdot 10^6$	14%
3	On(?b, ?c) OnTable(?b) ¬(?b.z ≤ 0.875) ∀?c.¬On(?c, ?b) ¬(?r.grip ≤ 0.5)	9351	100%

Abstract plans:	Refinable?
[Stack b2 on b3, Stack b1 on b2]	✗
...	
[Pick b2, Stack b2 on b3, Pick b1, Stack b1 on b2]	✗
...	
[Pick b4, Pick b2, Stack b2 on b3, Pick b1, Stack b1 on b2]	✗
...	
[Pick b4, PutOnTable b4, Pick b2, Stack b2 on b3, Pick b1, Stack b1 on b2]	✓

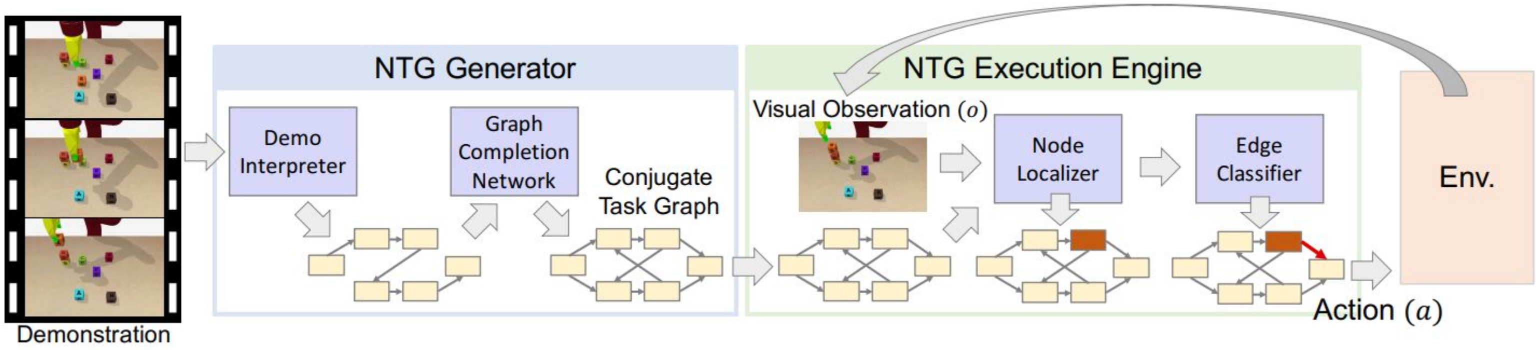
GNN-based Planner



Neural Task Graphs: Generalizing to Unseen Tasks from a Single Video Demonstration (CVPR 2019)

GNN-based Planner

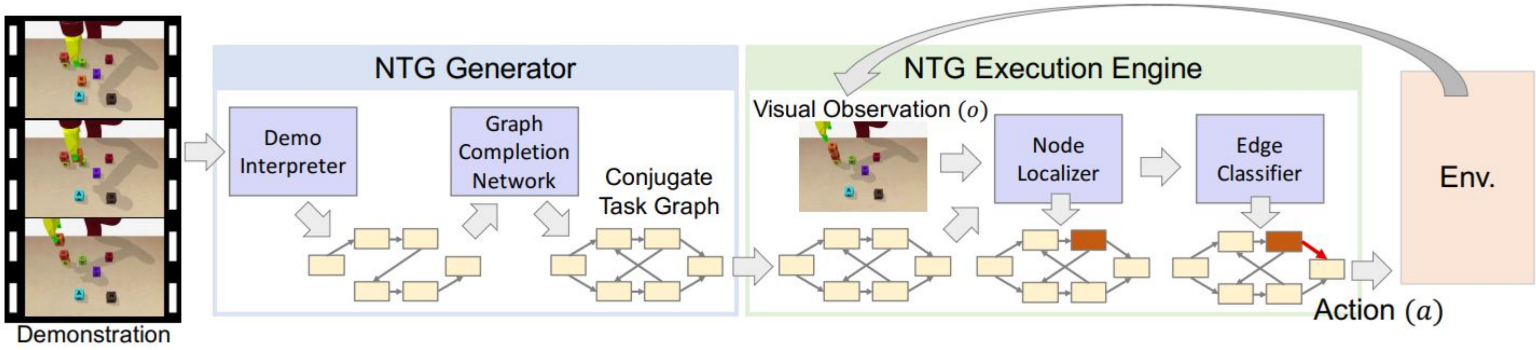
- Input: Video of expert Demonstration
- Output: Task Plan



Neural Task Graphs: Generalizing to Unseen Tasks from a Single Video Demonstration (CVPR 2019)

GNN-based Planner

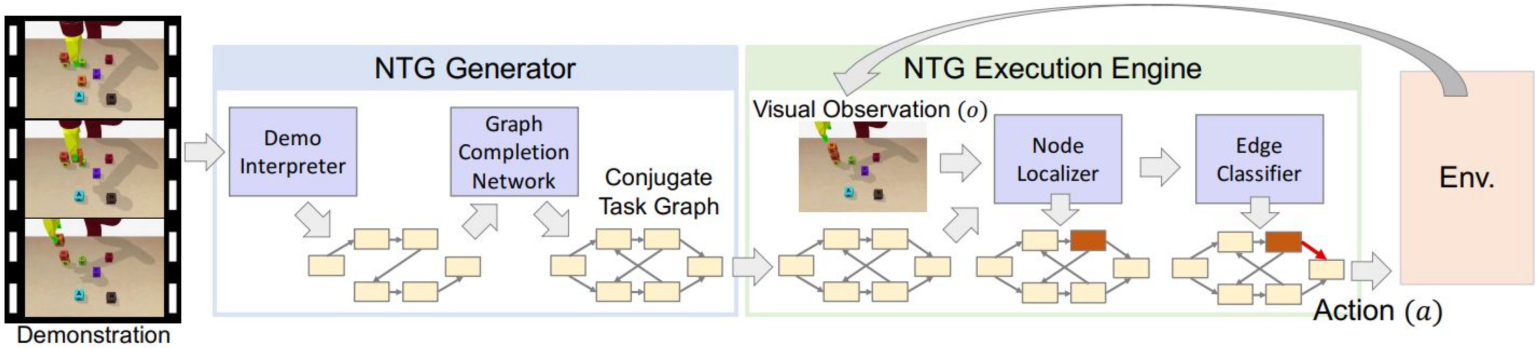
- Task graph: edges represent states, nodes represent actions
- This composition limits the number of nodes in the graph by considering only actions seen in the video



Neural Task Graphs: Generalizing to Unseen Tasks from a Single Video Demonstration (CVPR 2019)

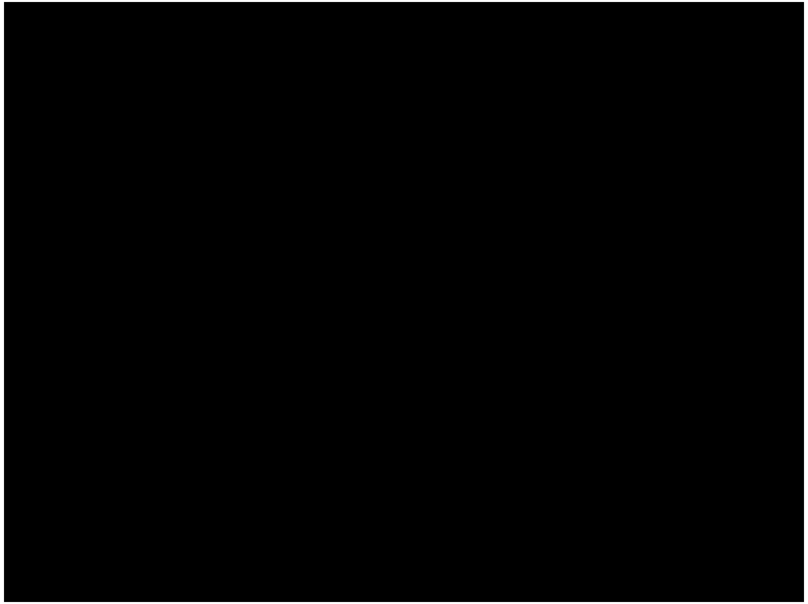
GNN-based Planner

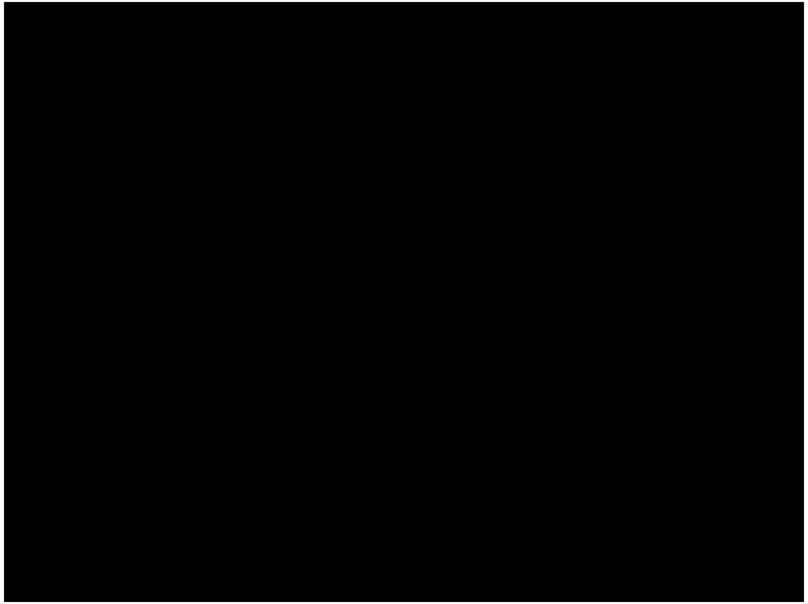
- The network learns a policy that creates one action for each observed time step of a video
- The graph completion network takes in a set of actions and learns the graph state transitions



Neural Task Graphs: Generalizing to Unseen Tasks from a Single Video Demonstration (CVPR 2019)

Hierarchical Reinforcement Learning



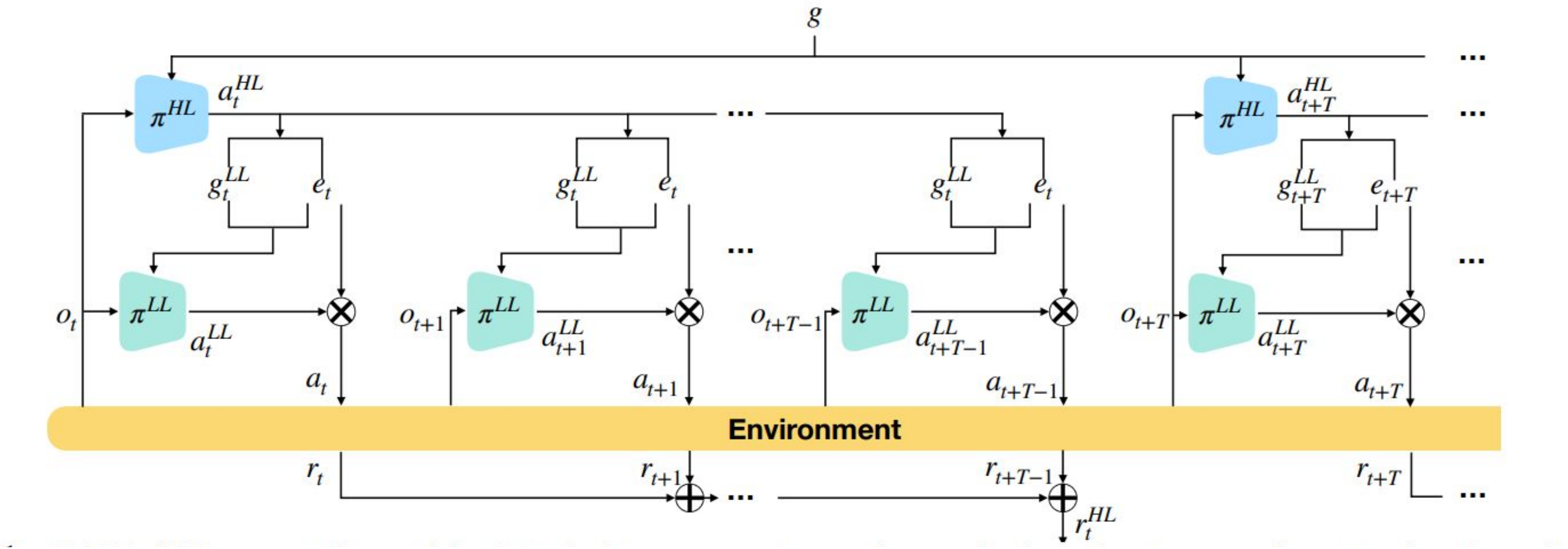


HRL4IN: Hierarchical Reinforcement Learning for Interactive Navigation with Mobile Manipulators (CoRL 2019)

Hierarchical Reinforcement Learning

- Uses multiple Policies
  - One learns the subgoals
  - Others are used to achieve these subgoals

Hierarchical Reinforcement Learning



HL: High Level, LL: Low Level, a: action, π: policy, g = goal, e = embodiment selector, r = reward

HRL4IN: Hierarchical Reinforcement Learning for Interactive Navigation with Mobile Manipulators (CoRL 2019)

Language in Robotics

Language in Robotics

Policy with Language Conditioning

Language-Conditioned Imitation Learning for Robot Manipulation Tasks
(Stepputtis et al. NeurIPS 2020)

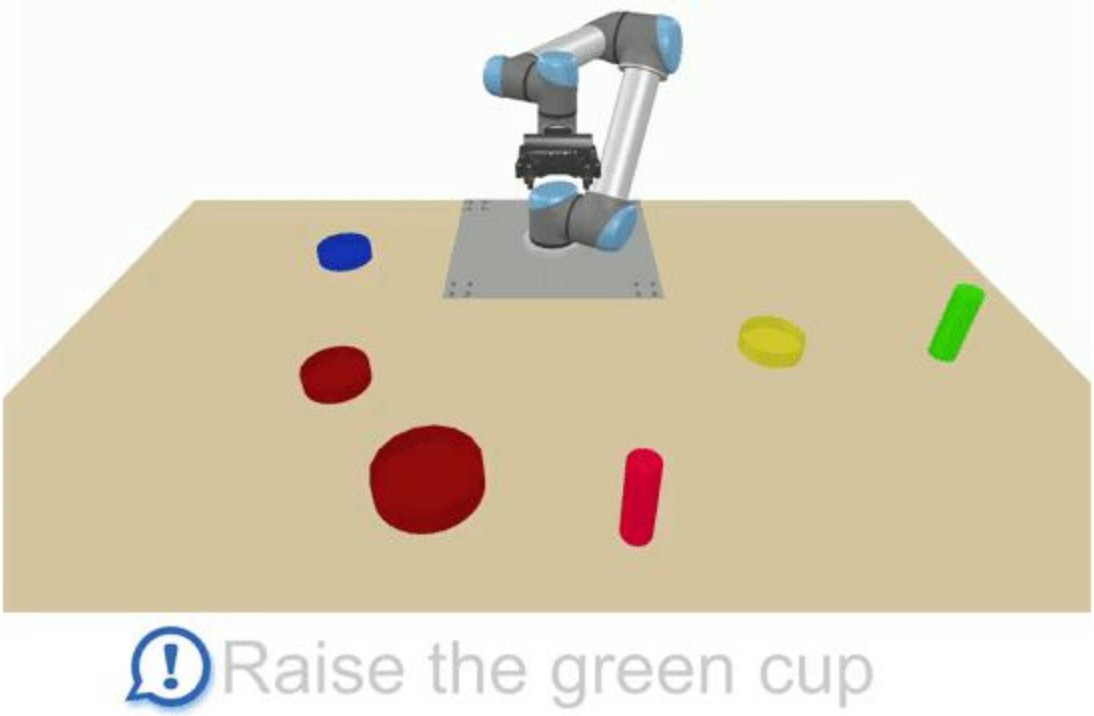
BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning
(Jang et al. CoRL 2021)

Open-World Object Manipulation using Pre-Trained Vision-Language Models
(Robotics@Google 2023)

Planning with Language Models

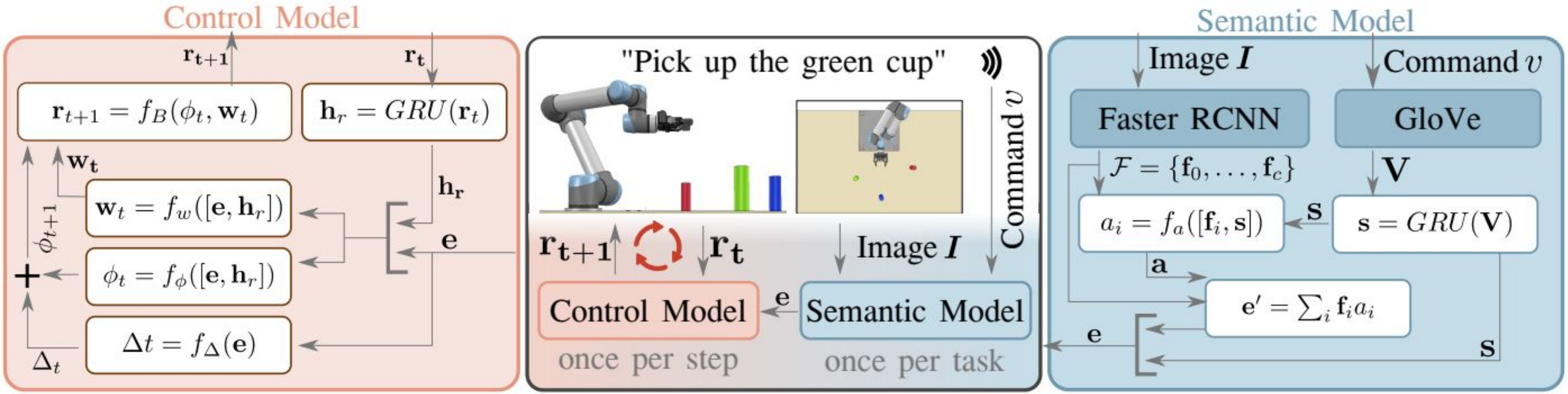
Do As I Can, Not As I Say:
Grounding Language in Robotic Affordances
(Ahn et al. CoRL 2022)

Language Conditioned Imitation Learning



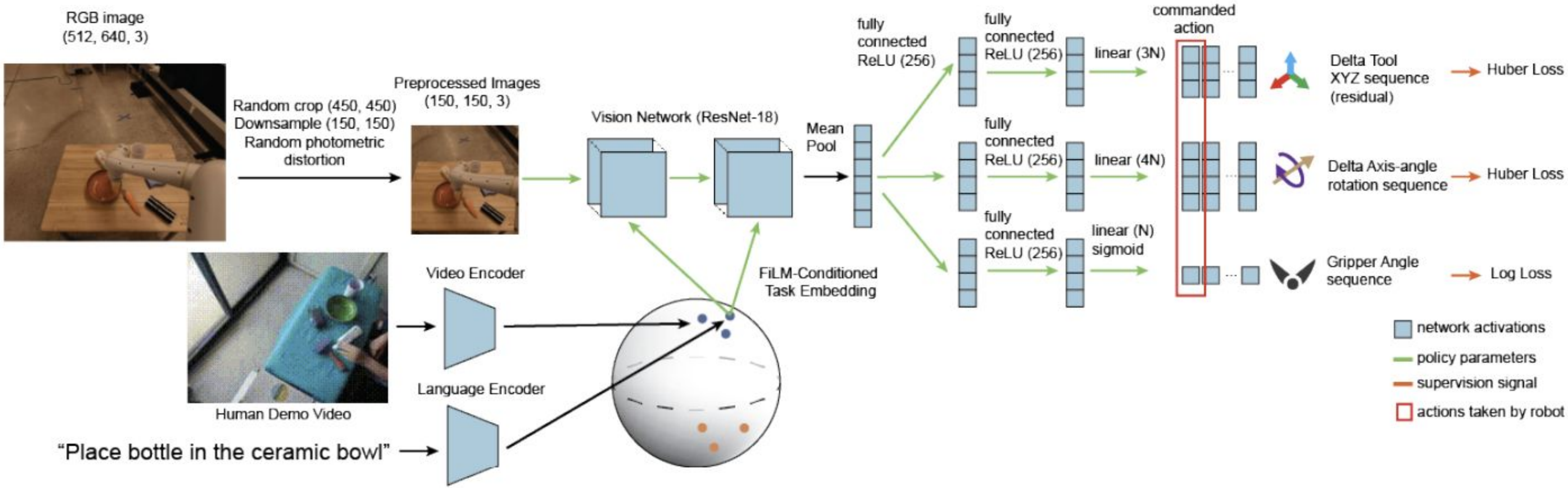
Language-Conditioned Imitation Learning for Robot Manipulation Tasks (Stepputtis et al. NeurIPS 2020)

Language Conditioned Imitation Learning



Language-Conditioned Imitation Learning for Robot Manipulation Tasks (Stepputtis et al. NeurIPS 2020)

Simpler End-to-end



BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning (Jang et al. CoRL 2021)

Simpler End-to-end

- Generalize to "Unseen Tasks"
- "Last-centimeter errors"

Skill	Held-out tasks (no demos during training)	Lang-conditioned (1 distractor)	Lang-conditioned (4-5 distractors)	Video-conditioned (4-5 distractors)
pick-place	'place sponge in tray'	83% (6.8)	82% (9.2)	22% (2.2)
	'place grapes in red bowl'	87% (6.2)	75% (10.8)	12% (7.8)
	'place apple in paper cup'	30% (8.4)	33% (12.2)	14% (7.8)
pick-wipe	'wipe tray with sponge'	40% (8.9)	0% (0)	28% (10.6)

Place Grapes in Ceramic Bowl

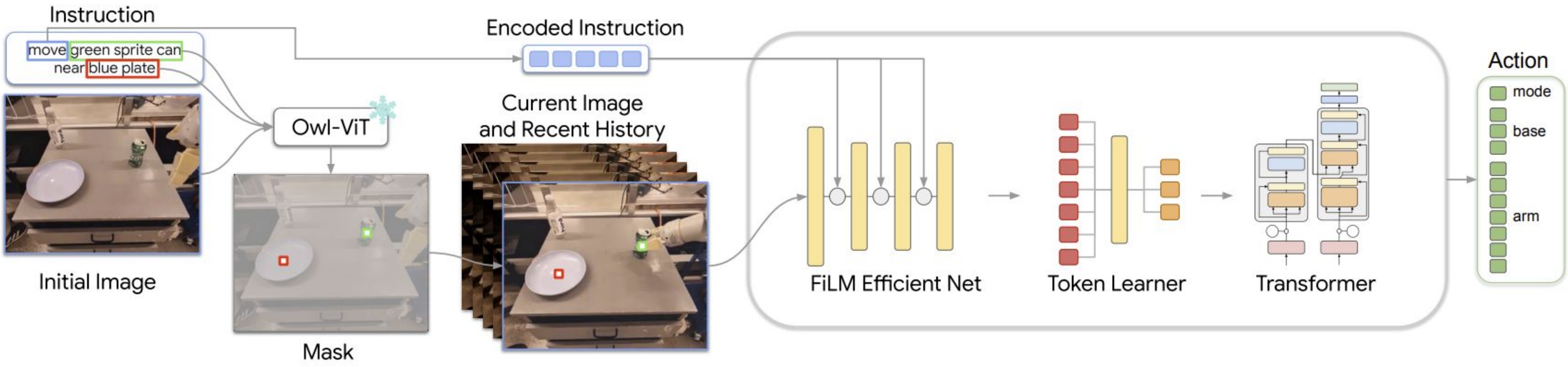
Place Bottle in Tray

Push Purple Bowl Across the Table

Wipe Tray with Sponge

BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning (Jang et al. CoRL 2021)

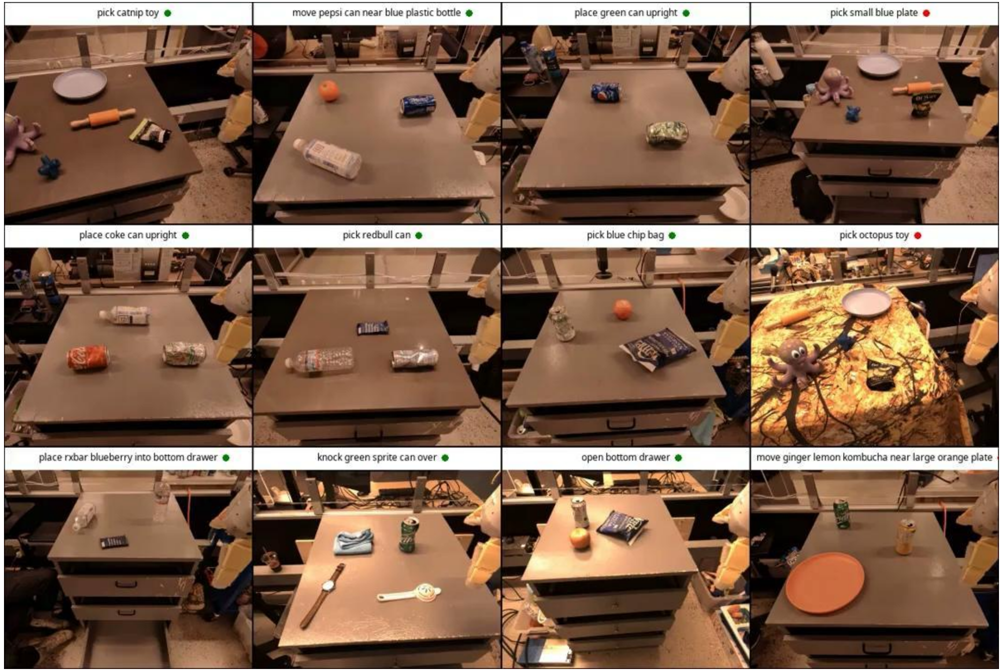
Vision-Language Models in Imitation



Open-World Object Manipulation using Pre-Trained Vision-Language Models (Robotics@Google 2023)

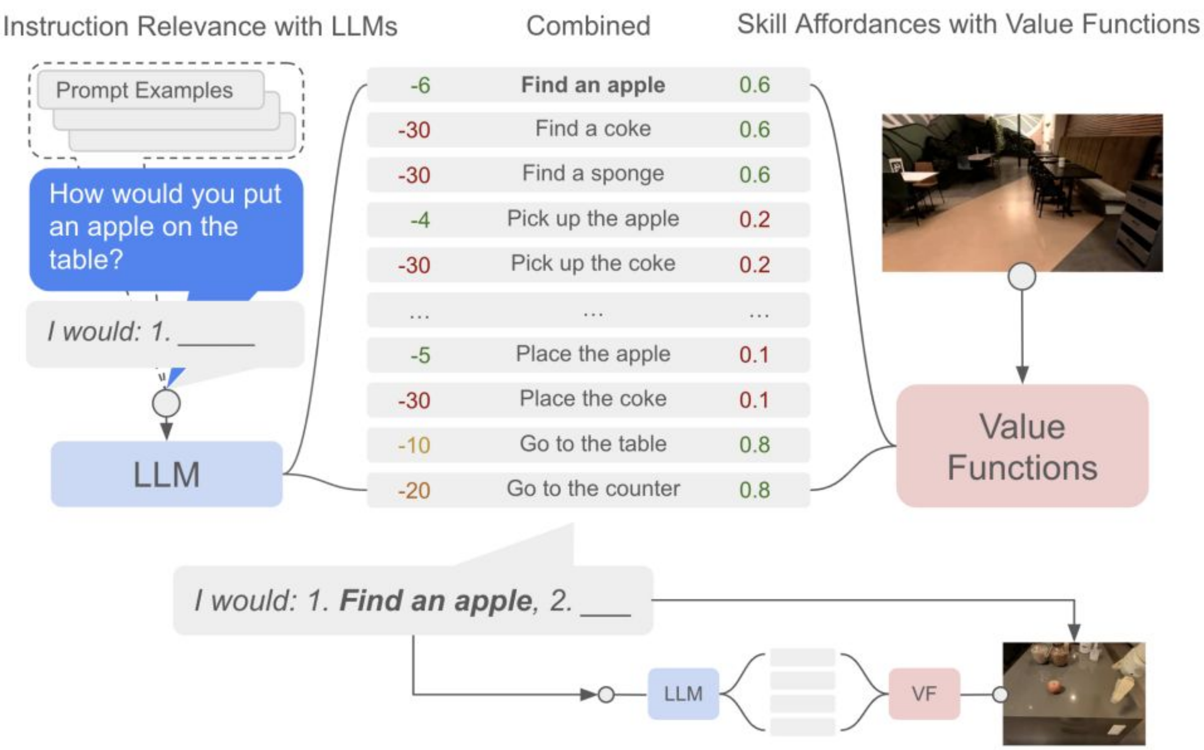
Vision-Language Models in Imitation

- Generalize to “Unseen Objects”



Open-World Object Manipulation using Pre-Trained Vision-Language Models (Robotics@Google 2023)

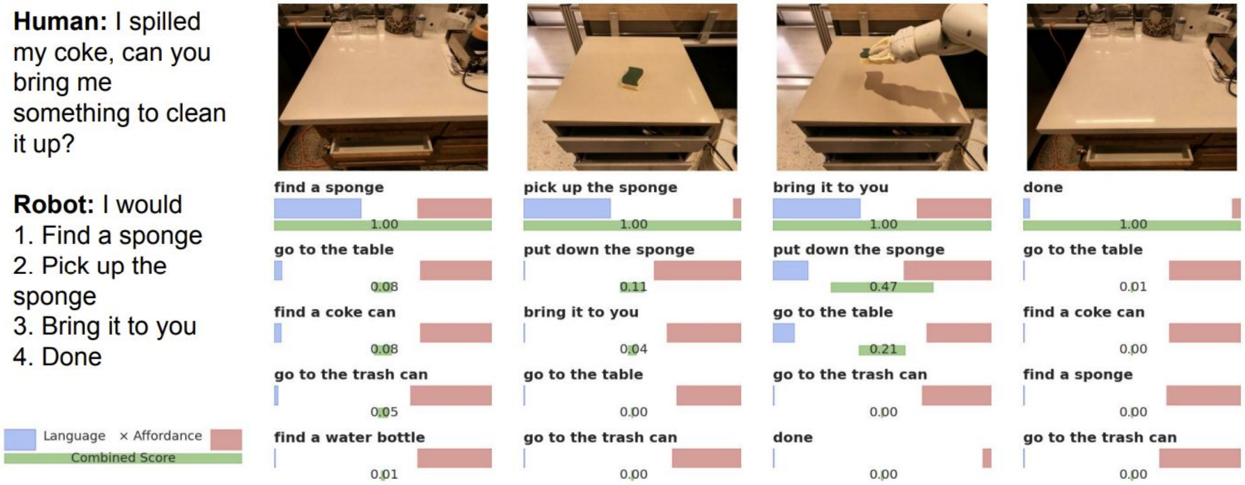
Robot Planning with Language Models



Do As I Can, Not As I Say: Grounding Language in Robotic Affordances (CoRL 2022)

Robot Planning with Language Models

- Train language-conditioned BC -> “Policy”
- Train language-conditioned RL -> “Value Function”
- Chain policies using LLM + Value



Do As I Can, Not As I Say: Grounding Language in Robotic Affordances (CoRL 2022)

Robot Planning with Language Models

Do As I Can, Not As I Say: Grounding Language in Robotic Affordances (CoRL 2022)

Summary

- Sense, plan, act framework
- Policies with algorithms
- Policies with neural networks
- Long horizon planning
- Language conditioned policies and planning

Thank you!