Explainable AI for Human-Robot Collaboration

Collaborative Artificial Intelligence and Robotics Lab

University of Colorado Boulder

Prof. Brad Hayes

Bradley.Hayes@Colorado.edu

http://www.cairo-lab.com/

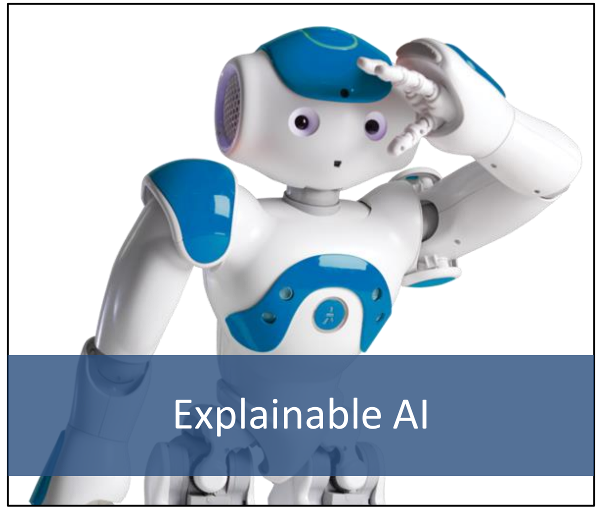
@hayesbh

http://bradhayes.info

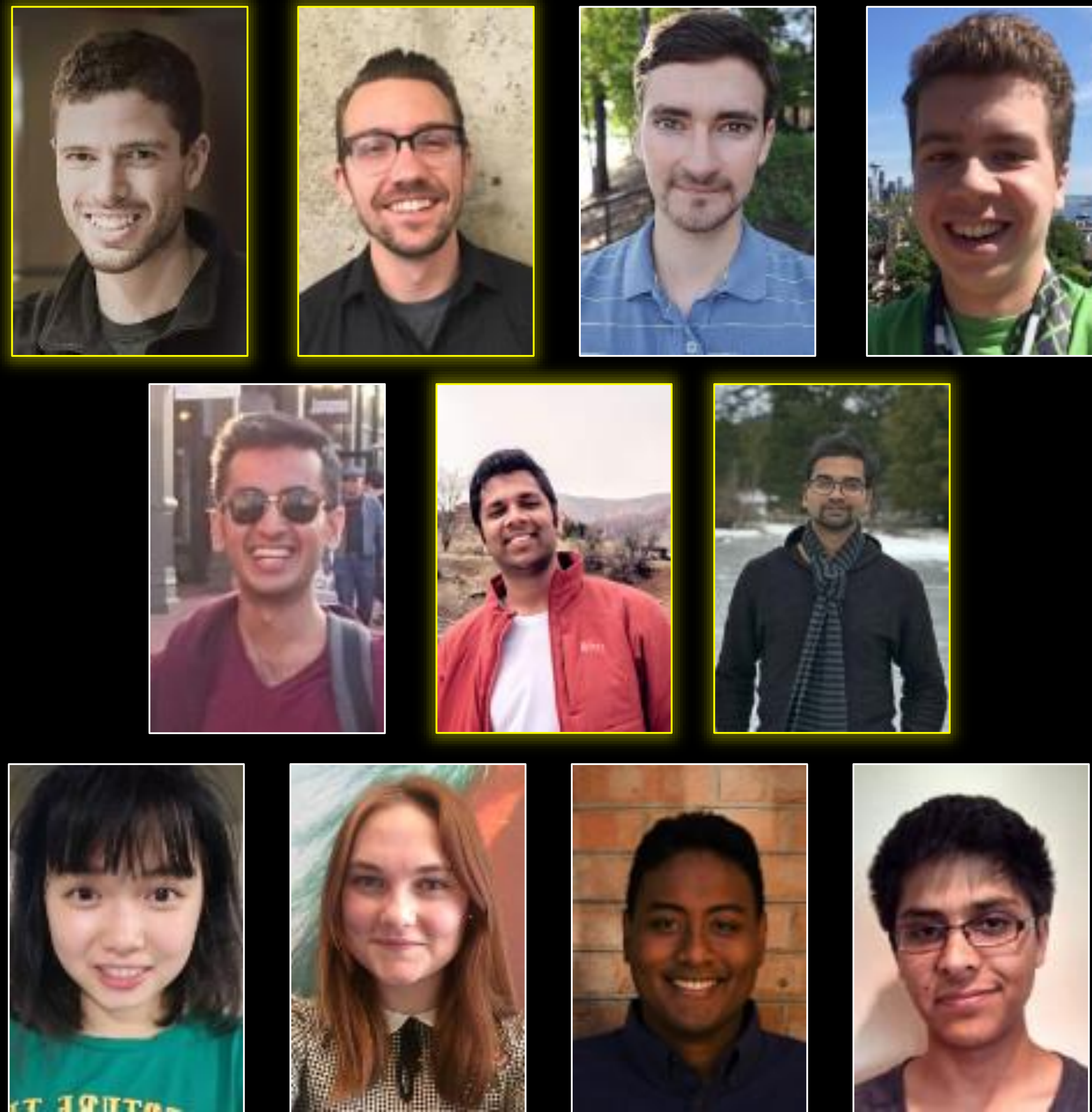
CAIRO

Collaborative AI and Robotics Laboratory

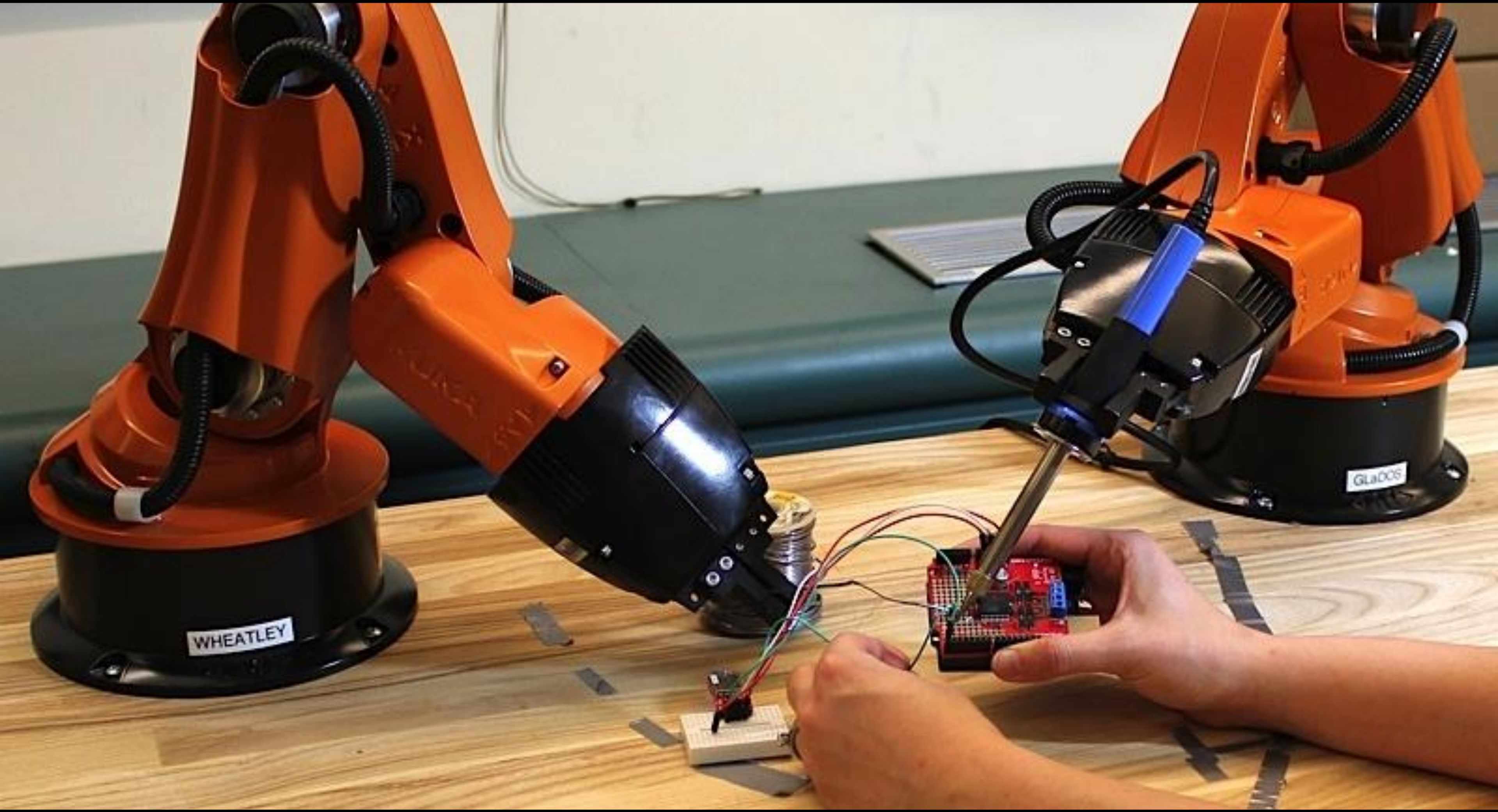
Research Themes



Collaborative AI and Robotics Laboratory



Collaborative Human-Robot Interaction



Human-in-the-loop artificial intelligence enables robot workers to make human collaborators **safer**, more **effective**, and more **efficient**.

So let's jump right in!

$$\theta^\star = \arg\max_\theta \underbrace{\sum_{t=1}^{T} E_{(\mathbf{s}_t,\mathbf{a}_t)\underset{\pi_\theta}{\sim} p_\theta(\mathbf{s}_t,\mathbf{a}_t)}[r(\mathbf{s}_t,\mathbf{a}_t)]}_{J(\theta)}$$

$$\underbrace{\pi_\theta(\mathbf{s}_1,\mathbf{a}_1,\ldots,\mathbf{s}_T,\mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[r(\tau)] = \int \pi_\theta(\tau)r(\tau)d\tau$$

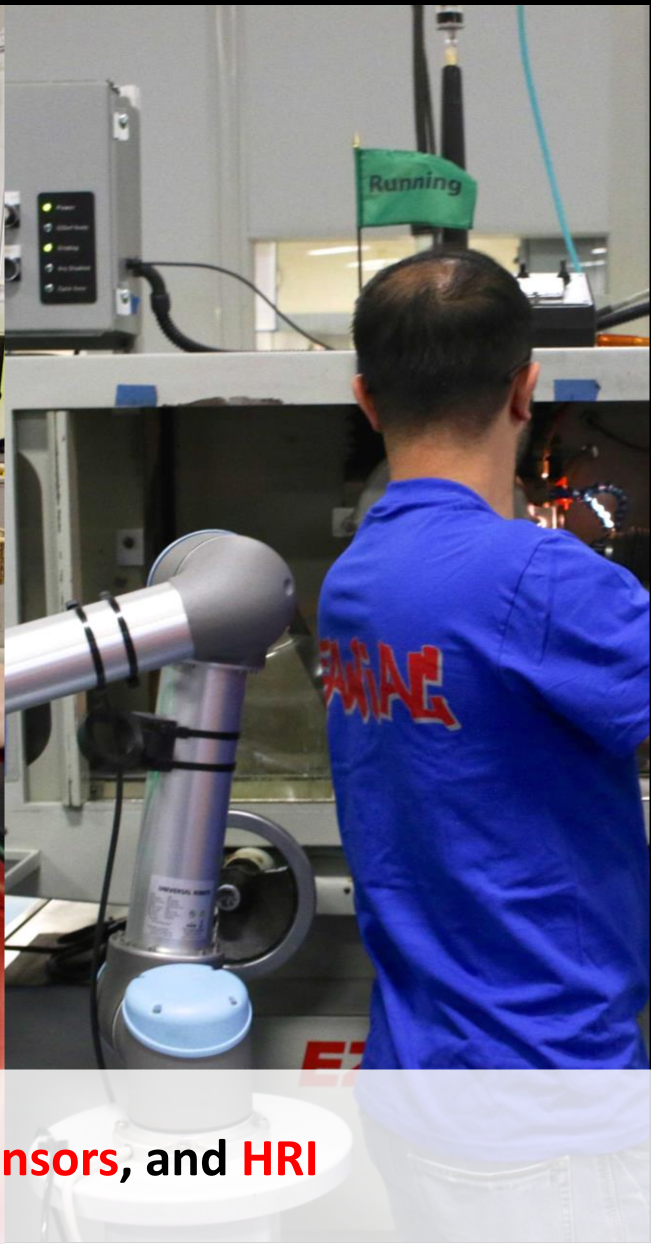
$$\pi_\theta(\tau)\nabla_\theta \log\pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta\pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta\pi_\theta(\tau)$$

$$\nabla_\theta J(\theta) = \int \nabla_\theta\pi_\theta(\tau)r(\tau)d\tau = \int \pi_\theta(\tau)\nabla_\theta\log\pi_\theta(\tau)r(\tau)d\tau = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta\log\pi_\theta(\tau)r(\tau)]$$

$$\boxed{\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T}\nabla_\theta\log\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_t,\mathbf{a}_t)\right)\right]}$$

$$\nabla_\theta \overbrace{\left[\log \cancel{p(\mathbf{s}_1)} + \sum_{t=1}^{T}\log\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log \cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}\right]}$$

Robot Co-workers



Cages are being replaced by algorithms, sensors, and HRI

Robot Co-workers

Robot Accidents in the Workplace

WHEN: AUGUST 2011
WHERE: BAKERY
WHAT HAPPENED: An employee was repairing a jammed conveyor belt in an oven when he became caught between a robotic arm and the belt. He was killed.

WHEN: MAY 2007
WHERE: PLASTICS FACTORY
WHAT HAPPENED: An employee was troubleshooting a robotic arm used to remove CD jewel cases when the arm struck the employee in his head and ribs. He died two weeks later.

WHEN: JULY 2006
WHERE: METAL FACTORY
WHAT HAPPENED: An employee was crushed between a robotic arm and the robot's work station. He appeared to have been reaching to remove a scrap the robot had dropped or to push the reset button, but there was no memory in the robot computer to know for sure. The employee was killed.

WHEN: MARCH 2006
WHERE: CAR FACTORY
WHAT HAPPENED: A robot caught an employee on the back of her neck and pinned her head between itself and the part she was welding. She was killed.

WHEN: DECEMBER 2001
WHERE: CAR FACTORY
WHAT HAPPENED: An employee was cleaning at the end of his shift and entered a robot's unlocked cage. The robot grabbed his neck and pinned the employee under a wheel rim. He was asphyxiated.

WHEN: AUGUST 1999
WHERE: METAL FACTORY
WHAT HAPPENED: A maintenance worker climbed a fence to repair a pin in a robot. It was still operating, and he became caught in the machine. He was killed.

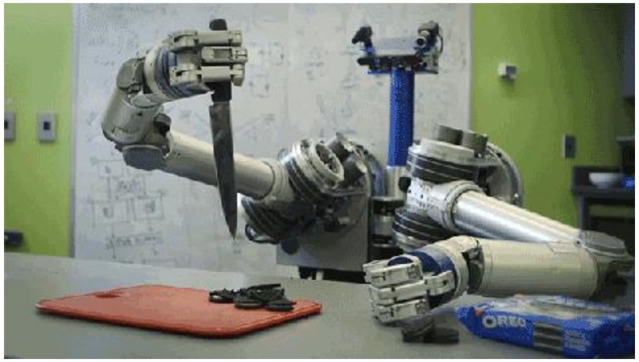
WHEN: NOVEMBER 1996
WHERE: SPORTING GOODS MANUFACTURER
WHAT HAPPENED: An employee was using a robot to weld and drill basketball backboards. When he noticed a half-done hole, he manually drilled it. The robot thought that meant the cycle was complete and unexpectedly turned, pinning the employee against the wall. He was hospitalized.

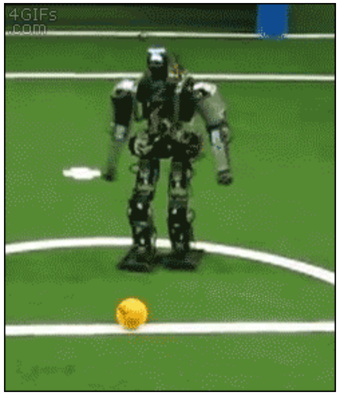
WHEN: JUNE 1999
WHERE: MEATPACKING PLANT
WHAT HAPPENED: An employee accidentally activated a robot when he stepped on a conveyor belt where robots were moving boxes of meat. He became trapped. When his co-workers removed the robot, he fell to the floor. He was killed.

WHEN: FEBRUARY 1996
WHERE: ALUMINUM FACTORY
WHAT HAPPENED: Three workers were watching a robot pour molten aluminum when the pouring unexpectedly stopped. One of them left to flip a switch to start the pouring again. The other two were still standing near the pouring operation, and when the robot restarted, its 150-pound ladle pinned one of them against the wall. He was killed.

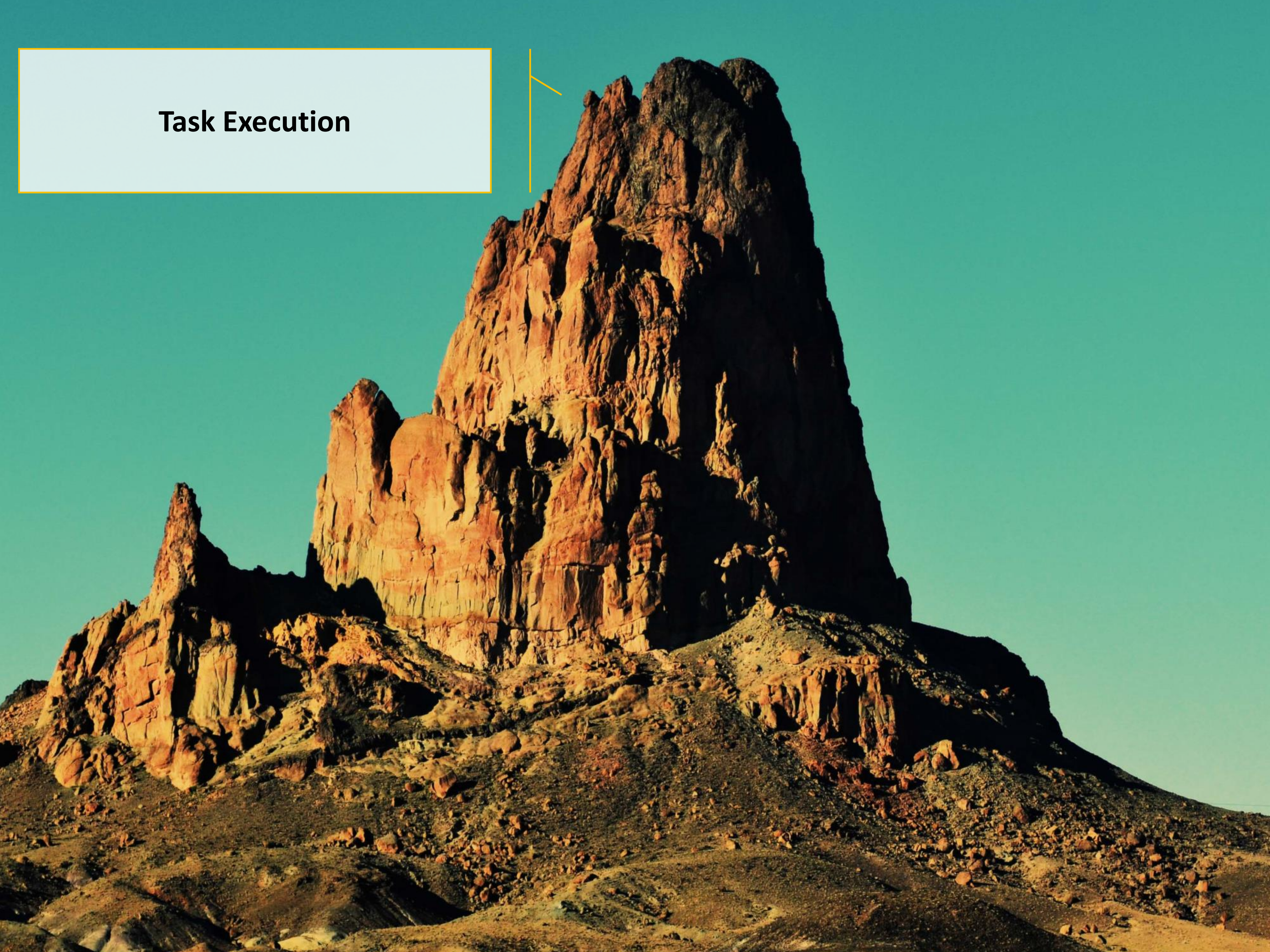
Robots are the future!

…but it's really hard to make them do what we want.





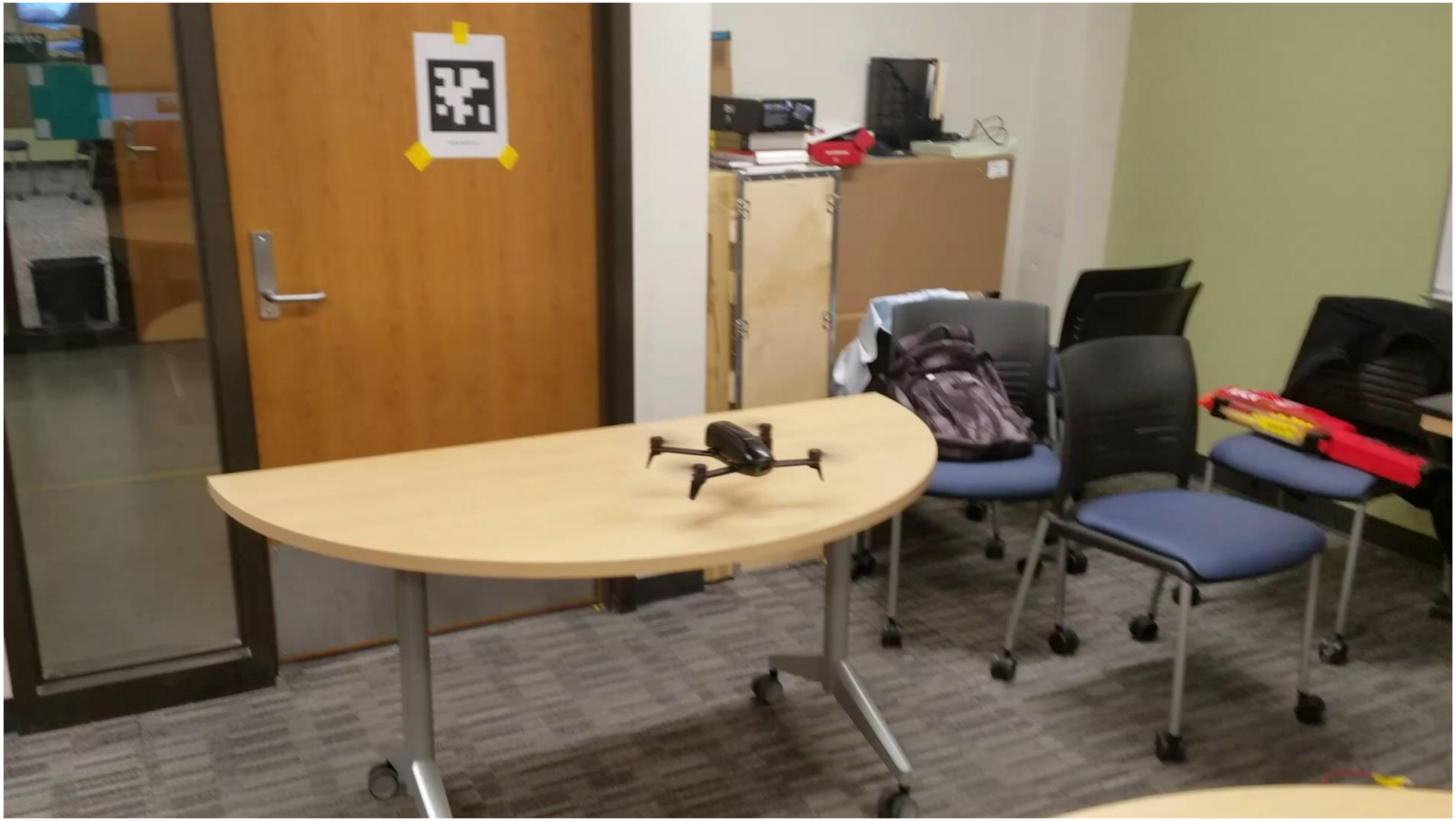
Task Execution



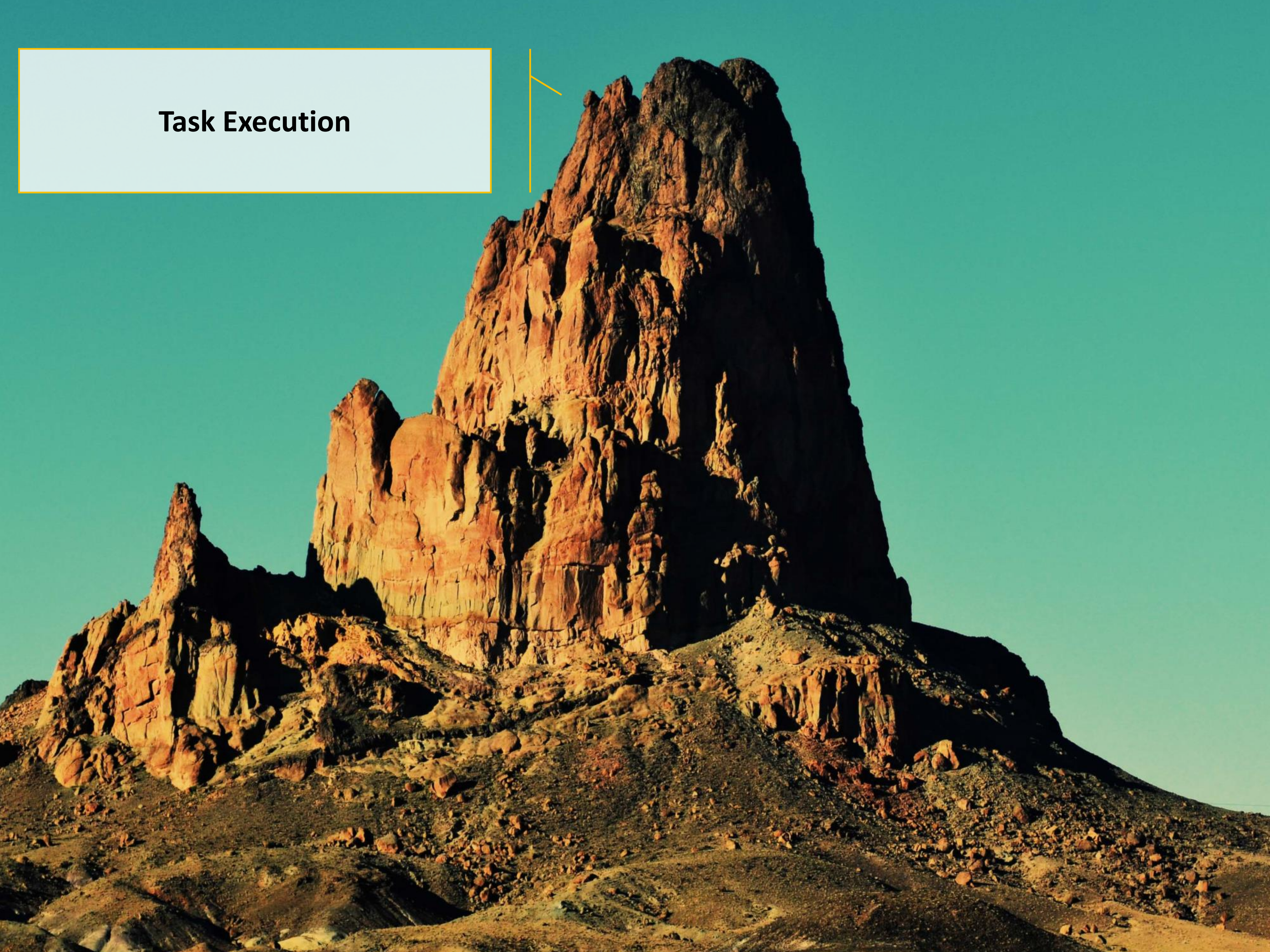
Robotics is Hard

Nobody knows everything

Even worse: HRI is multi-disciplinary



Task Execution

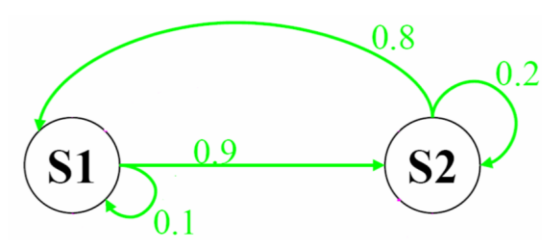


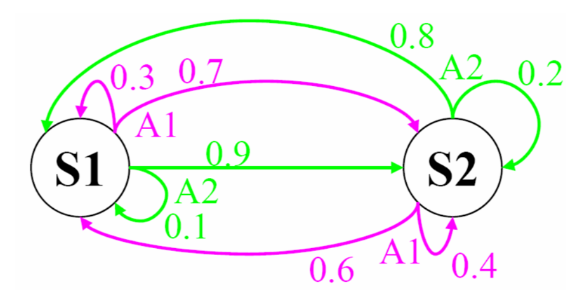
Markov Model Chart

Do we have control over the state transitions?
(Are we picking which actions are executed)

Are the states completely observable?

	NO	**YES**
YES	Markov Chain	MDP
NO	HMM	POMDP





Difficulty of Human-Robot Collaboration

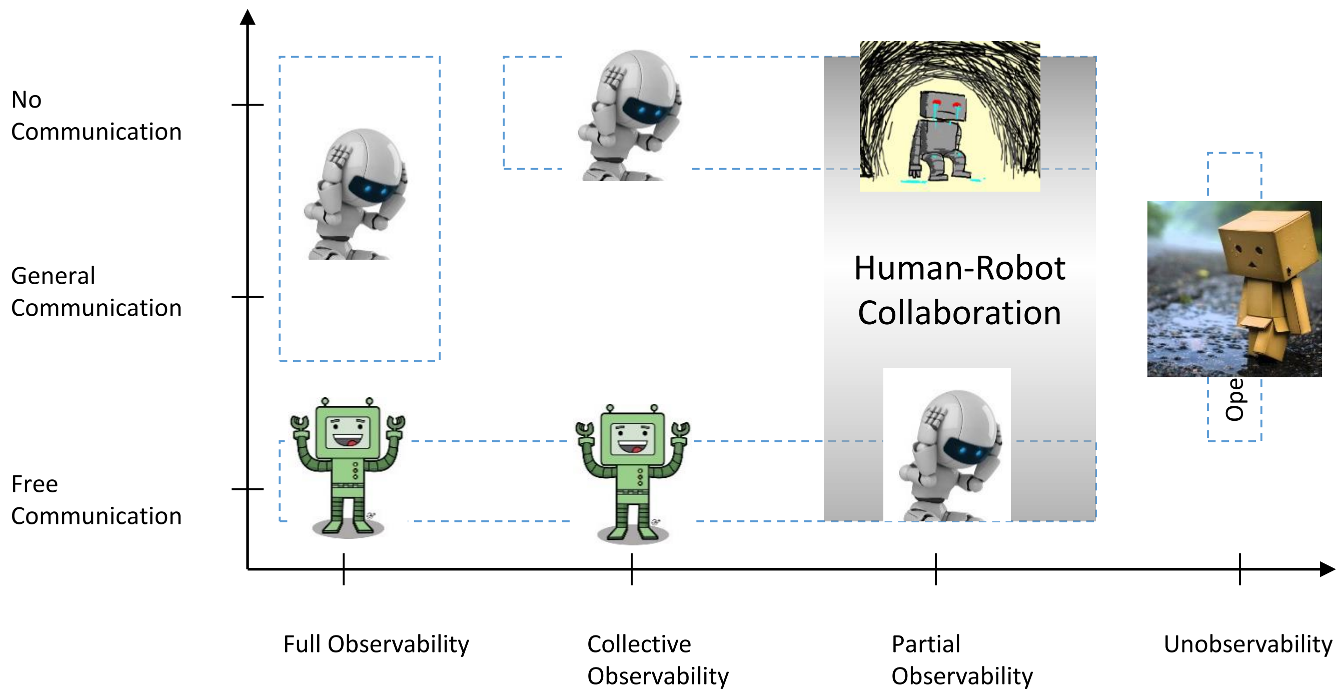
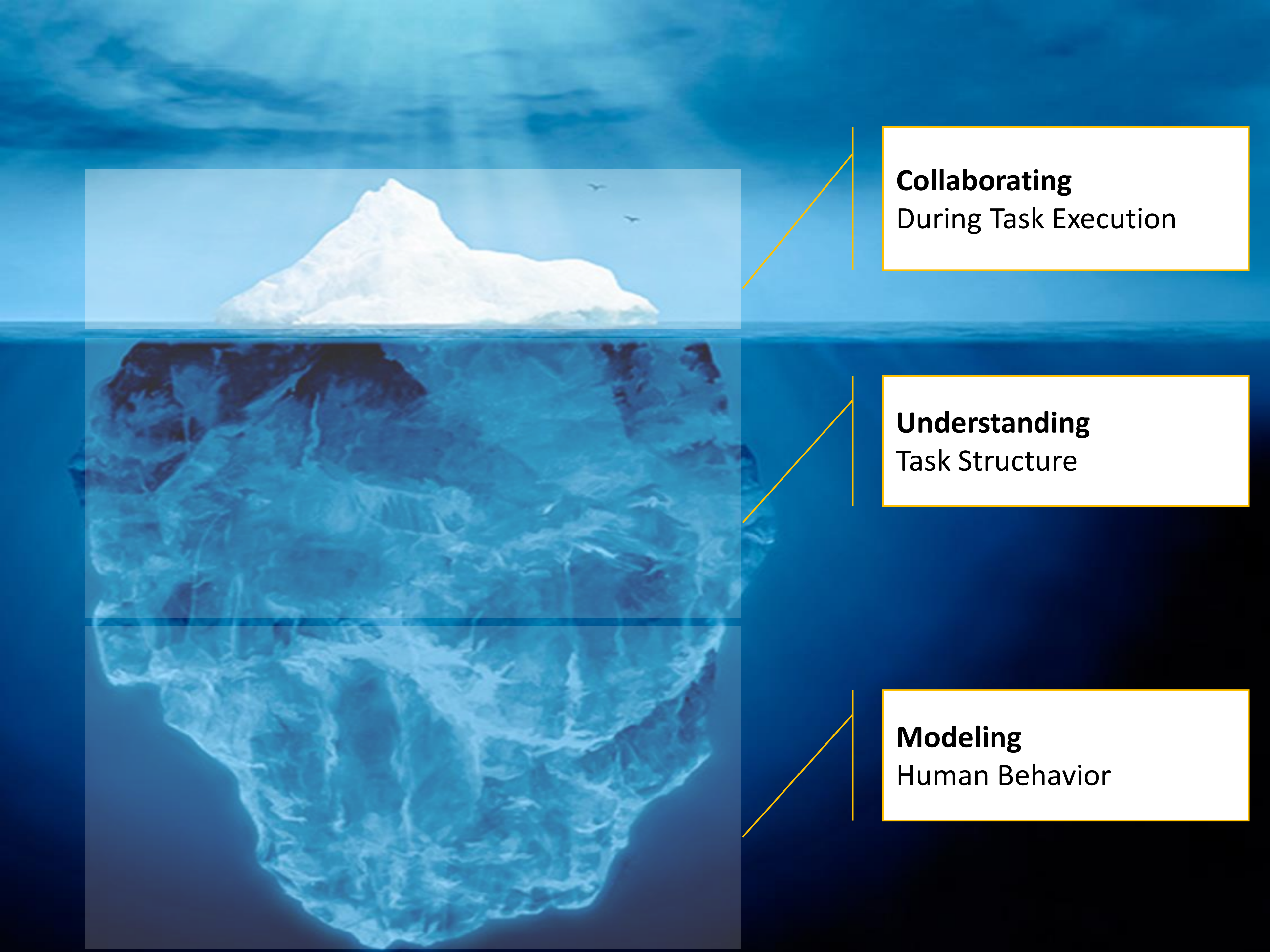


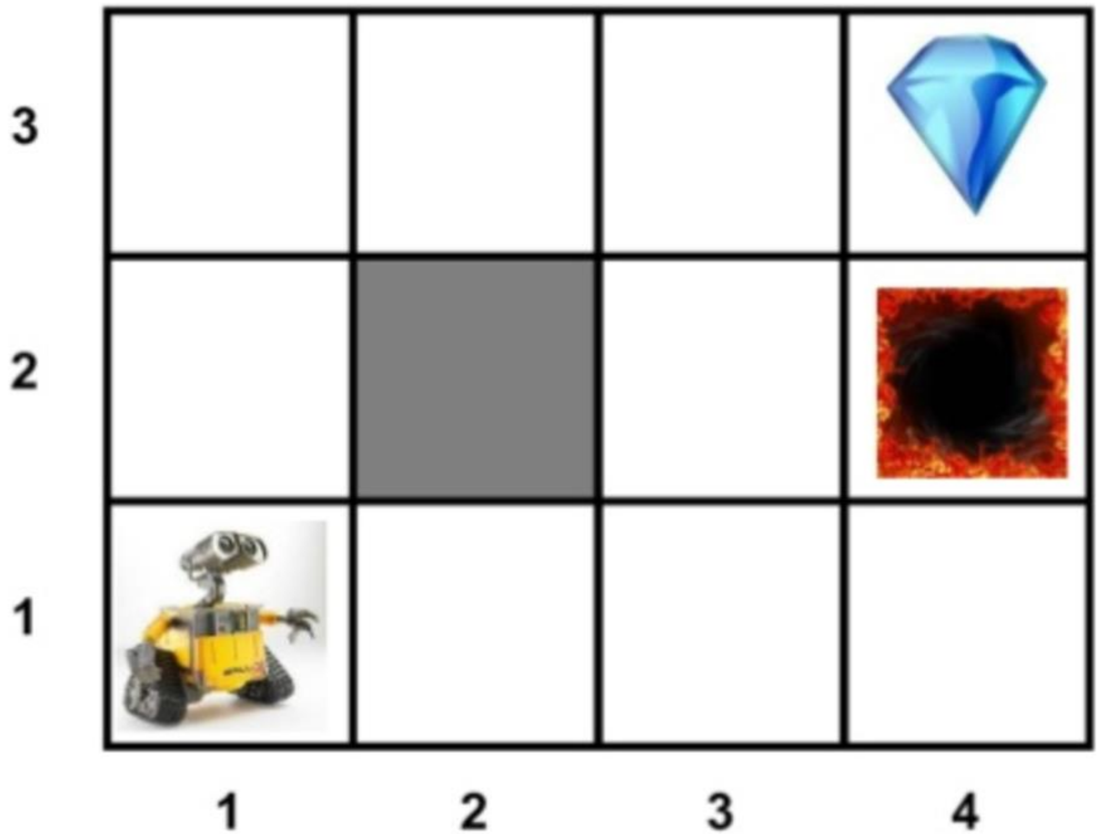
Chart credit: Maayan Roth, “Markov Model for Multi-Agent Coordination”


Collaborative Task Execution

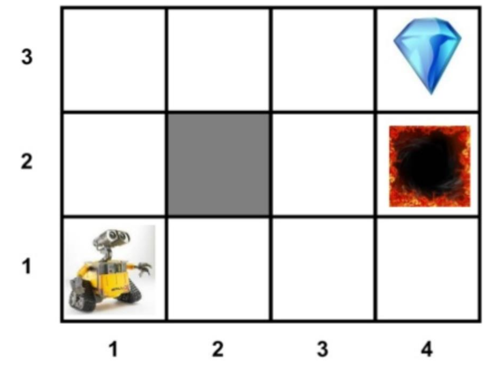

Collaborating
During Task Execution
Yikes :(


Collaborating
During Task Execution
Understanding
Task Structure
Modeling
Human Behavior

Sample Problem



Sample Problem Terminology



A **state** is a representation of the world

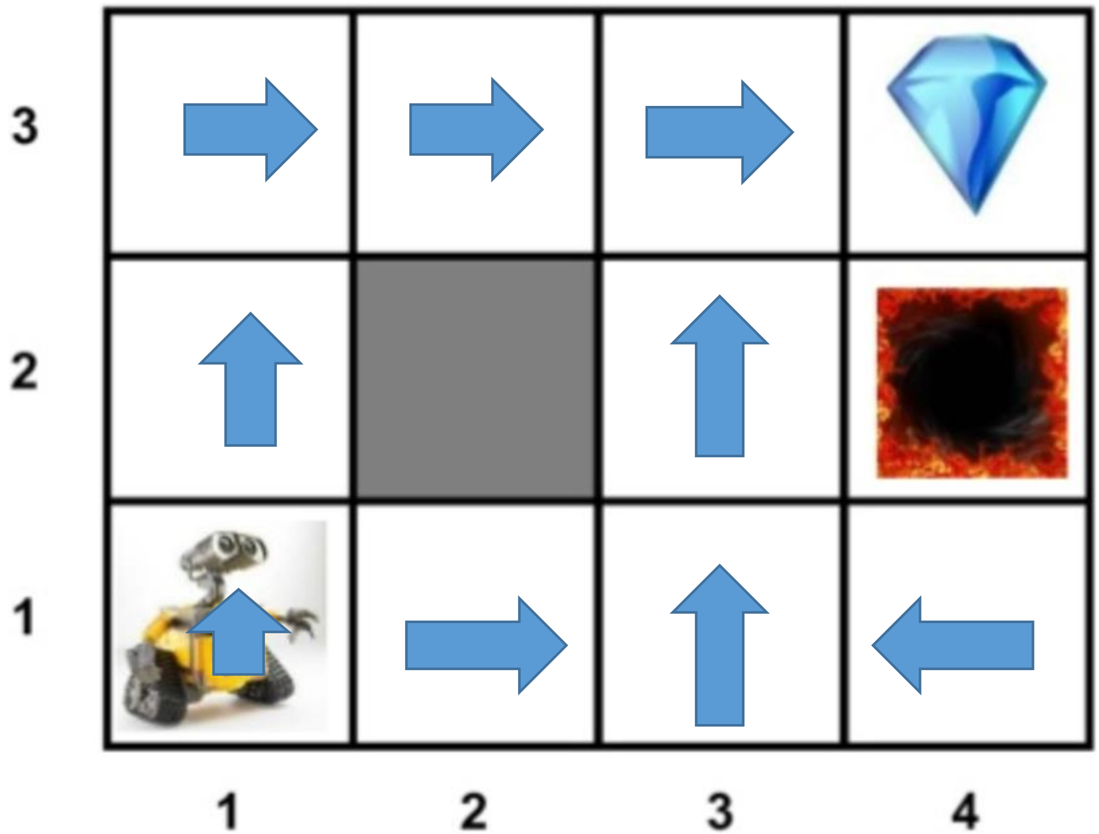
An **action** is something that transitions you from one state to another (can also be a self-transition!)

A **transition function** T(s,a,s') provides the probability that a particular action **a** taken in a particular state **s** will bring the system to state **s'**

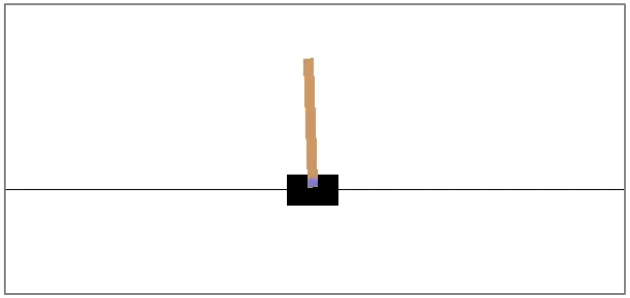
A **reward function** R(s, a) provides the value of taking a particular action **a** in state **s**

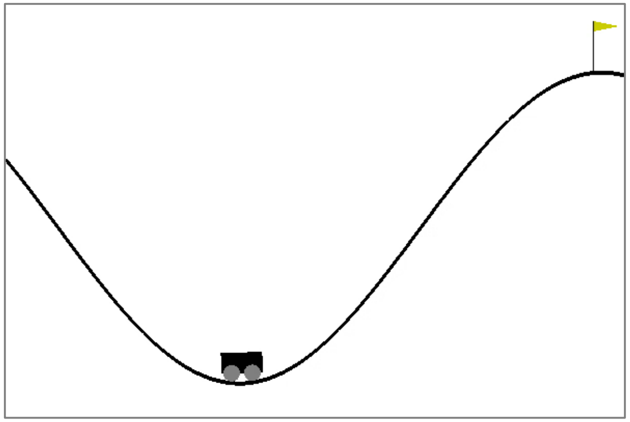
Sample **Policy**

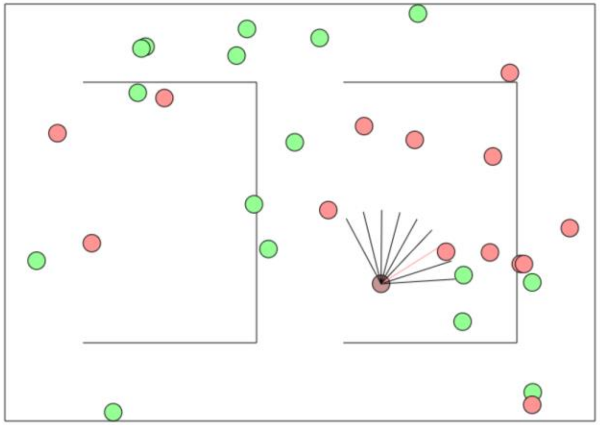
$\pi: S \rightarrow A$

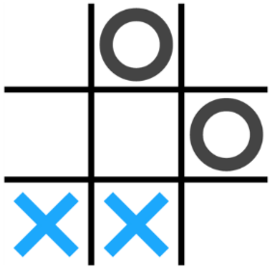


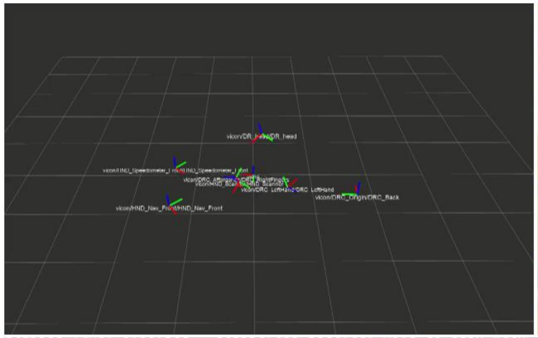
State Representation is Critical

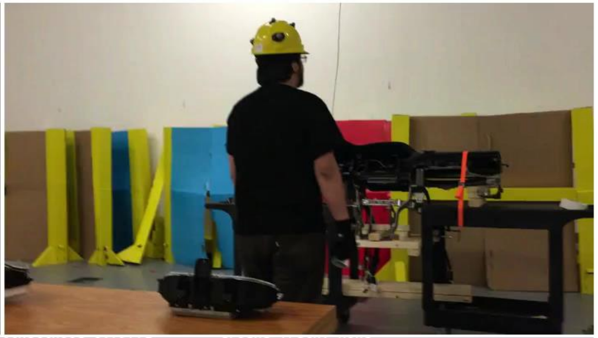










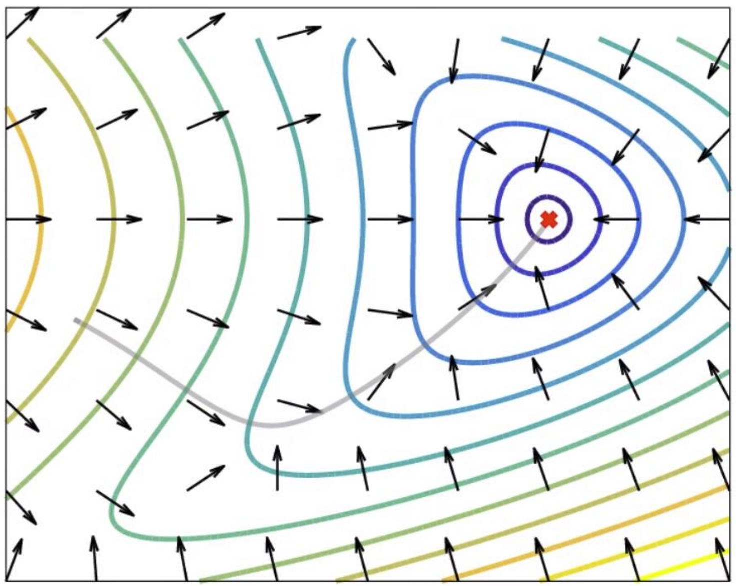


```
Elapsed Time: 0.1sec      Classified activity move_to_dash with likelihood 0.84128      Ground Truth: None
Elapsed Time: 0.13sec     Classified activity move_to_dash with likelihood 0.84811      Ground Truth: None
Elapsed Time: 0.17sec     Classified activity move_to_dash with likelihood 0.86419      Ground Truth: None
Elapsed Time: 0.2sec      Classified activity move_to_dash with likelihood 0.867        Ground Truth: None
Elapsed Time: 0.23sec     Classified activity move_to_dash with likelihood 0.95099      Ground Truth: None
```

Motion Planning & Optimal Control

Optimal Control: Finding the best control policy for a desired goal

Closed-Loop Solutions

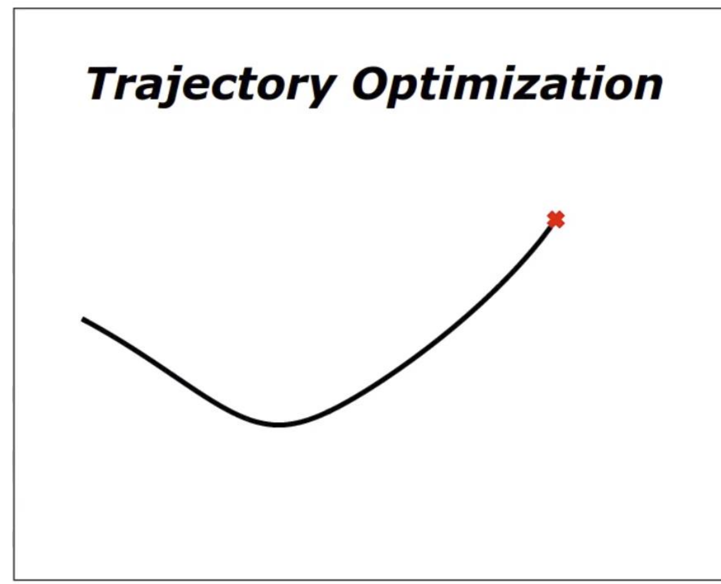


$$u = u(x)$$

"Global Method": Gives action at all states

Very expensive to compute

Open-Loop Solution



$$u = u(t)$$

"Local Method": Gives action at relevant states

Usable in high dimensions

Trajectory Optimization:

Problem Statement

- Trajectory $\xi: t \in [0, T] \rightarrow C$ *Maps time to configurations*

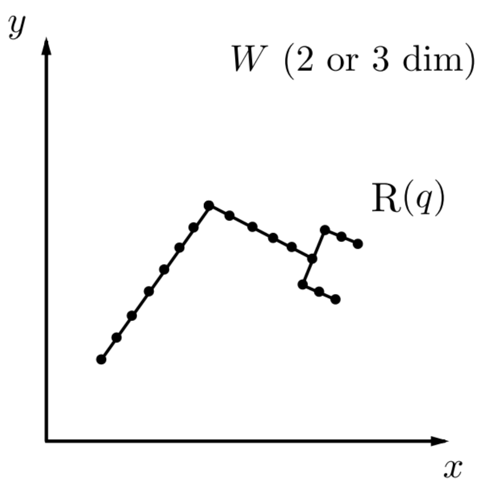
- Objective Functional $U: \Xi \rightarrow \mathbb{R}^+$ *Maps trajectories to scalars*

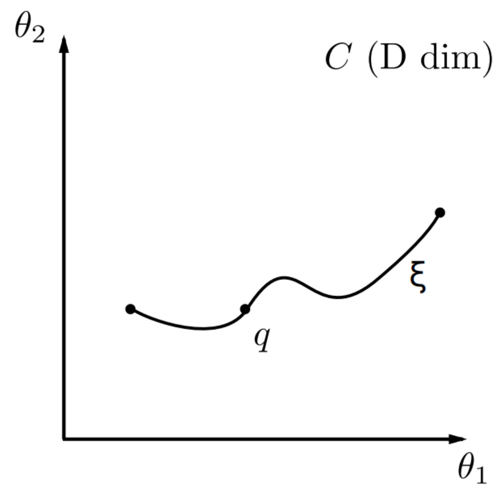
Set of possible trajectories

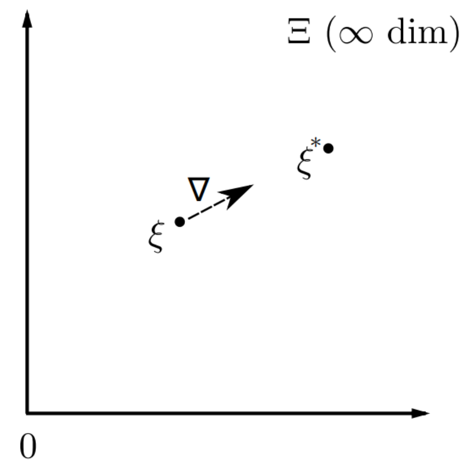
- The objective U encodes traits we want our path to have
  - Path length
  - Efficiency
  - Obstacle avoidance
  - Legibility
  - Uncertainty reduction
  - Human comfort

Goal: Leverage the benefits of randomized sampling with asymptotic optimality

Problem Specification: Spaces







World Space W($\mathbb{R}$^3)

Configuration Space
C($\#DoF$)

Trajectory Space
$\Xi(\infty\ dim)$

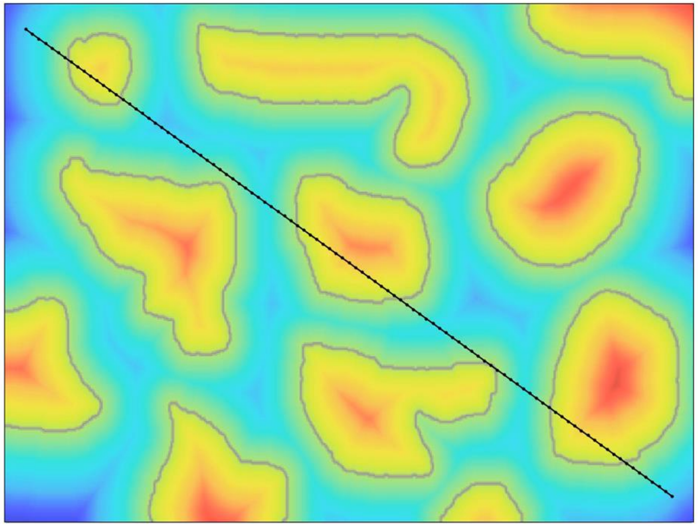
Robot pose in World Space (set of points) → Single point in Configuration Space

Trajectory through Configuration Space (set of points) → Single point in Trajectory Space

Problem Specification: Optimization

Trajectory Optimization seeks to find an optimal trajectory ξ^*:

$$\xi^* = argmin_{\{\xi \in \Xi\}} U[\xi]$$

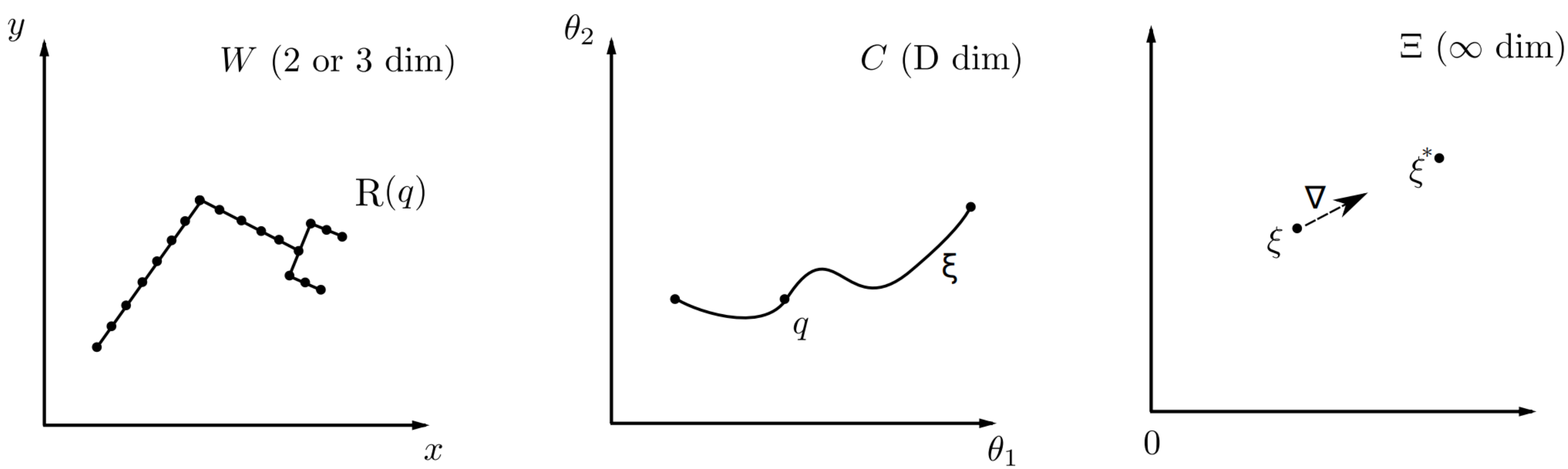


s.t. $\xi(0) = q_s$

$\xi(T) = q_g$

…(any other constraints we want)

Problem Specification: Optimization



Want to optimize ξ to a global minimum of our objective **U**

=> Usually too hard!

Instead, optimize ξ to a local minimum of our objective **U**

=> If the solution is bad, resample ξ and try again

Donald Michie's criteria for Machine Learning (ML)

Weak criterion:

ML occurs whenever a system generates an updated basis building on sample data for improving its performance on subsequent data.

Strong criterion:

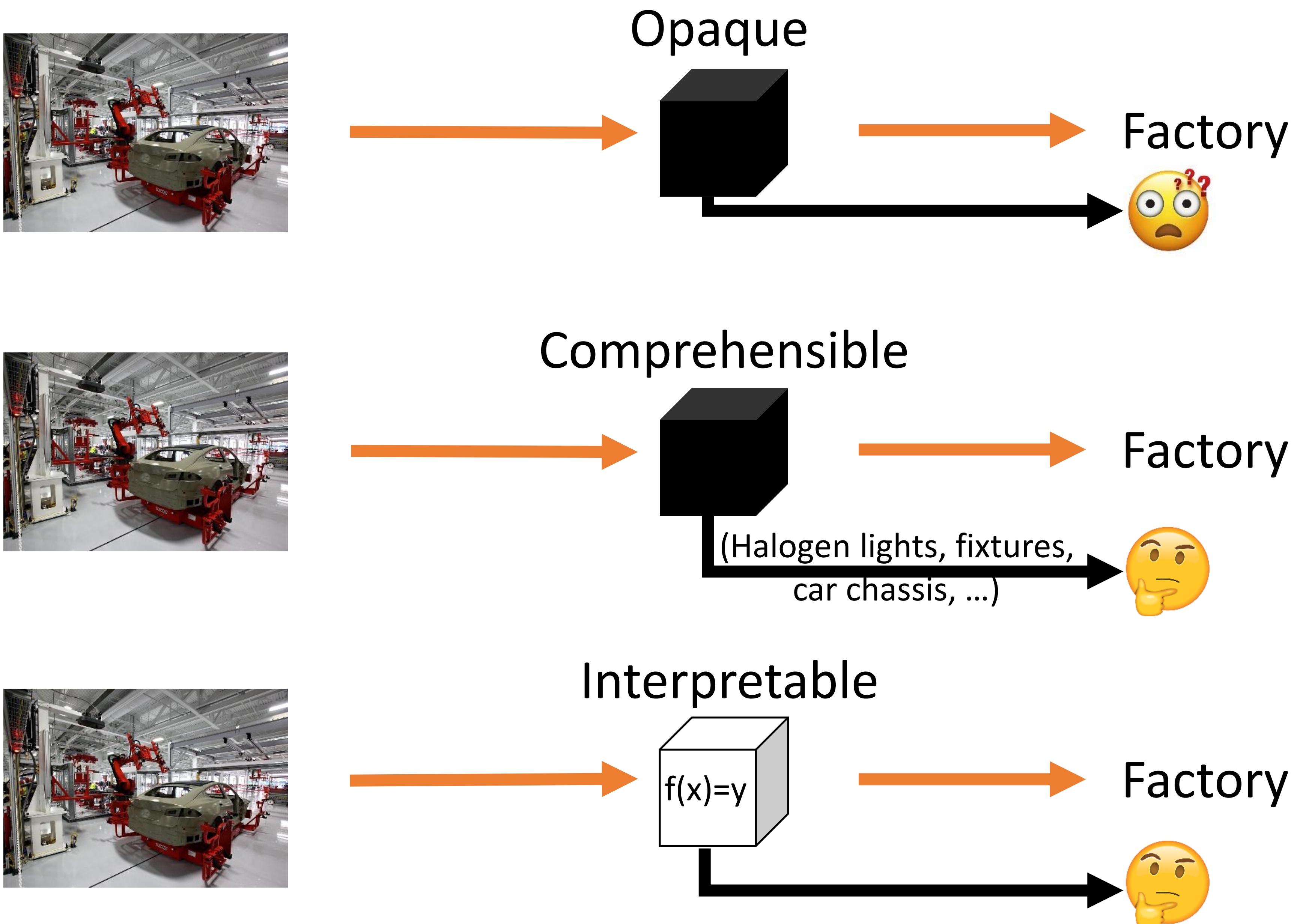
Weak criterion + ability of system to communicate internal updates in explicit symbolic form.

Ultra-strong criterion:

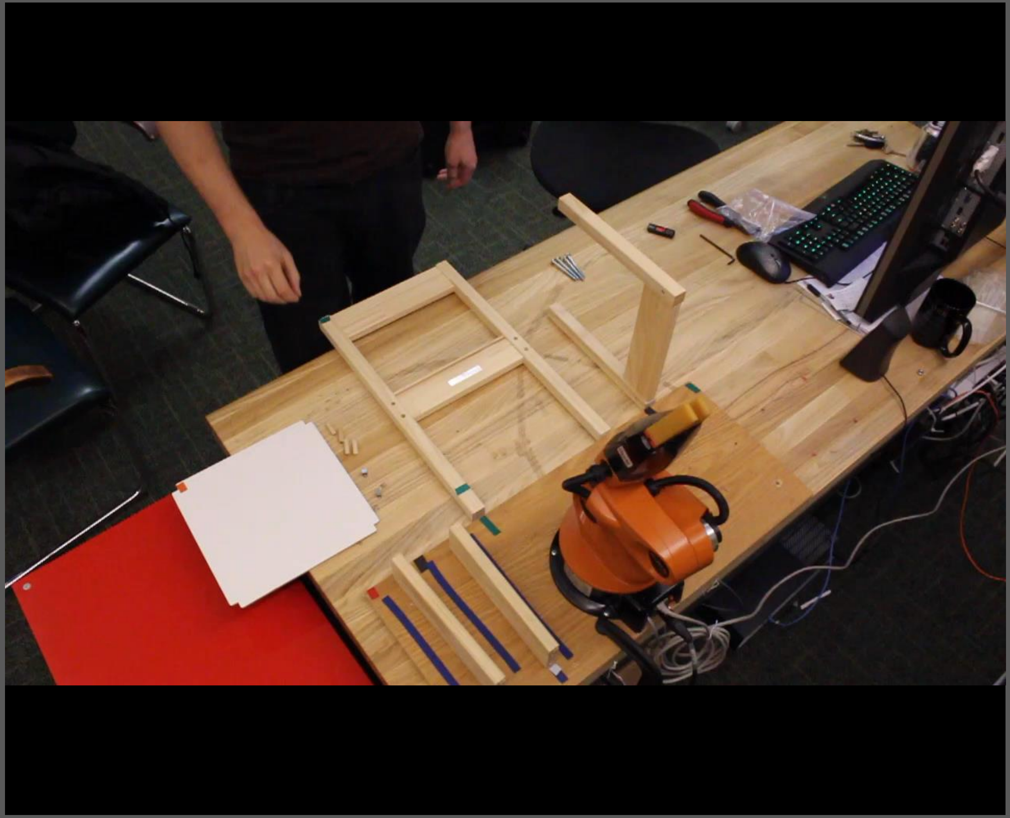
Strong criterion + communication of updates must be operationally effective (i.e. user is required to understand updates and consequences should be drawn from it).

Where is this?

Relating Different Types of Systems

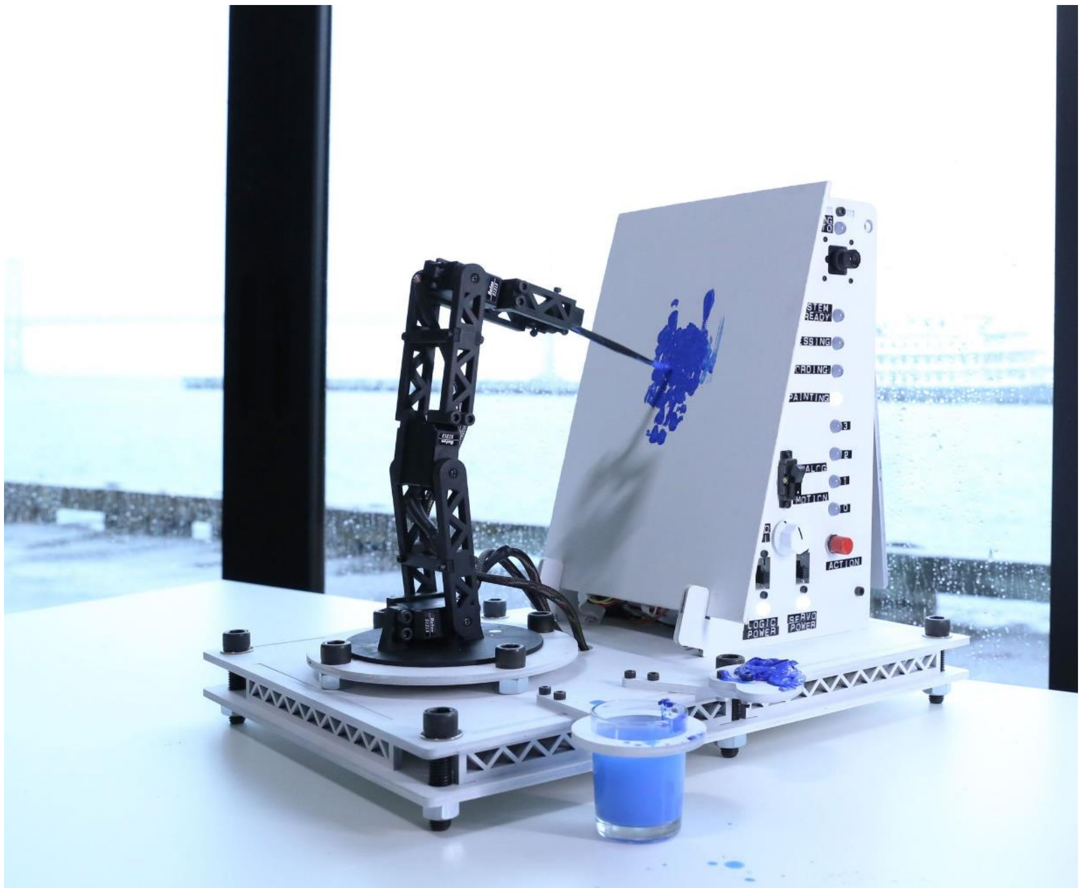


Let’s Make a Furniture-Building Collaborative Robot



Let's unpack this problem…

Consider the following challenge

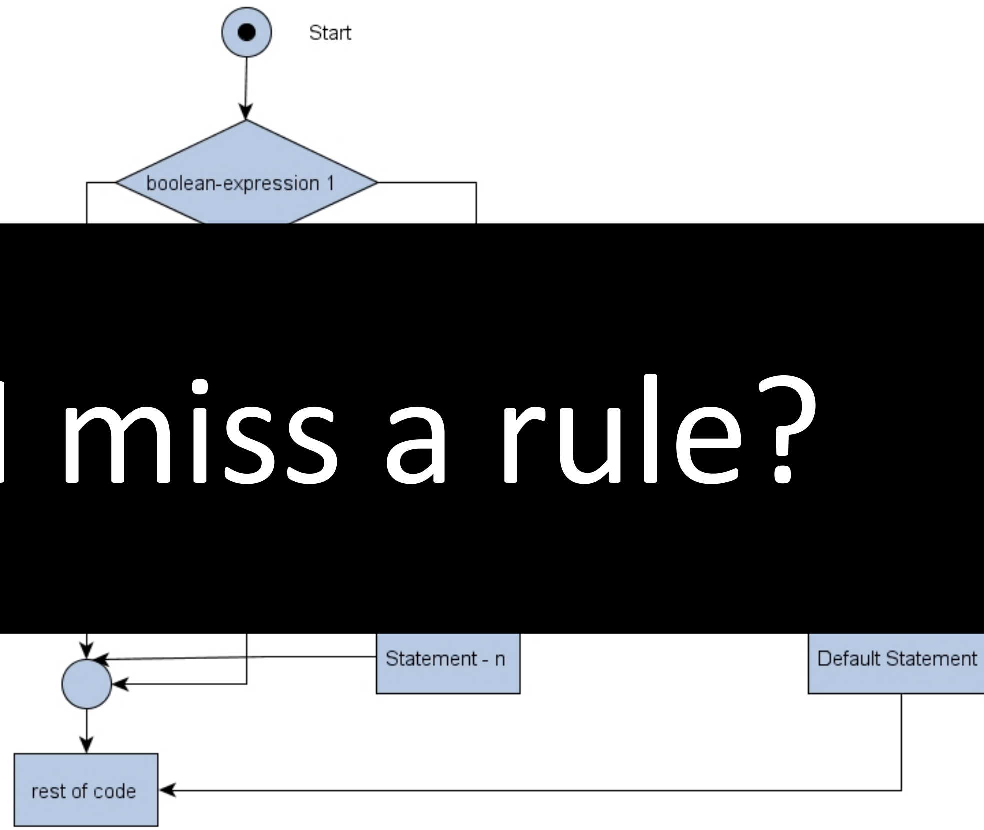


- How do we get a robot to write for us?
- What’s the best way to encode the actions the robot has to perform?
- How can we teach the robot to draw a single letter properly?

Painstakingly Program Each Motion

- We can code each motion one at time, giving the motors set amounts to move at each step of the process
- This is brittle! What if the robot isn't in the exact same spot as it was when we programmed it?

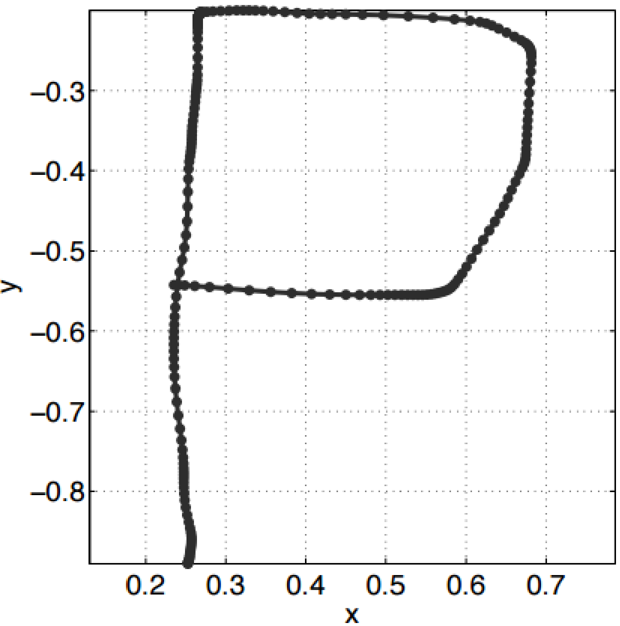
Add hand-written rules and logic!



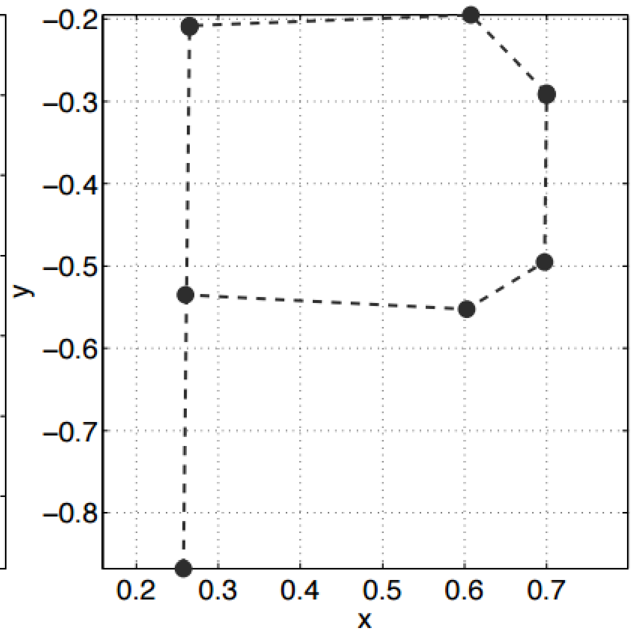
Else-if Ladder statement flow chart

What if I miss a rule?

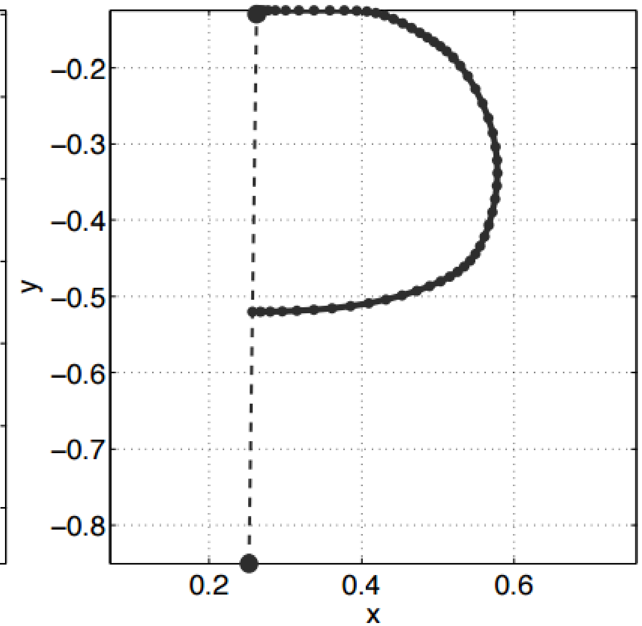
Learning from Demonstration



Trajectory
Demonstration



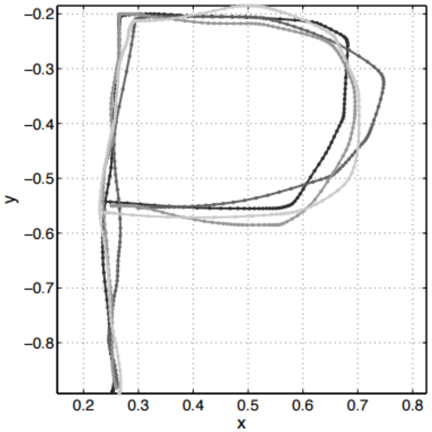
Keyframe
Demonstration



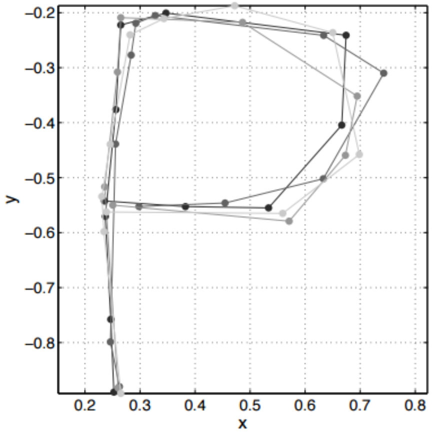
Hybrid
Demonstration

Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective
B Akgun, M Cakmak, JW Yoo, AL Thomaz

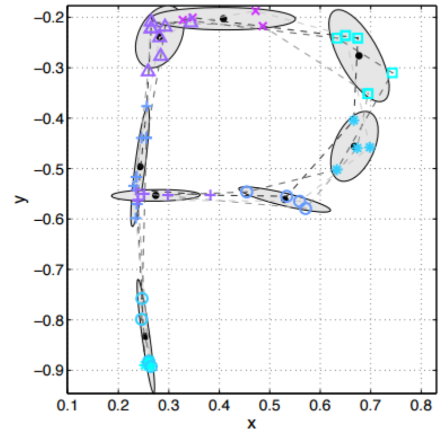
Learning to Draw “P” from Examples:



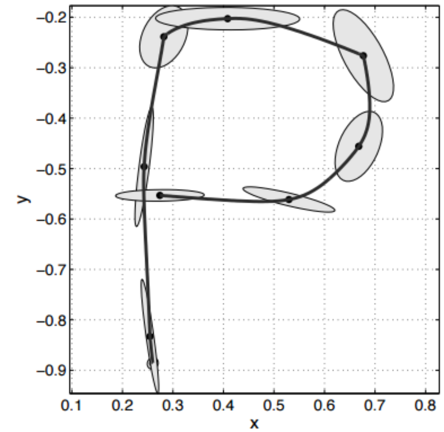
Continuous trajectories in 2D



Data converted to keyframes

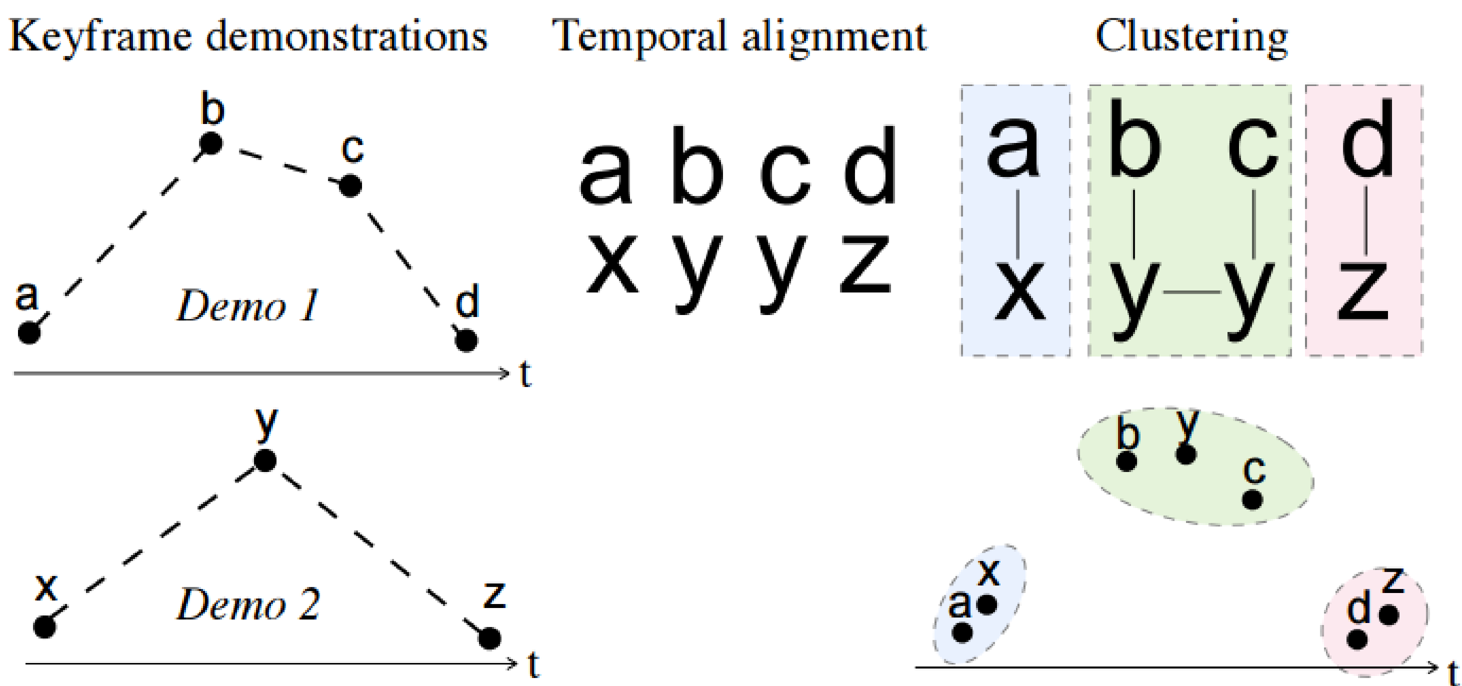


Clustering of keyframes and the sequential pose distributions



Learned model trajectory

Dealing with variations in speed

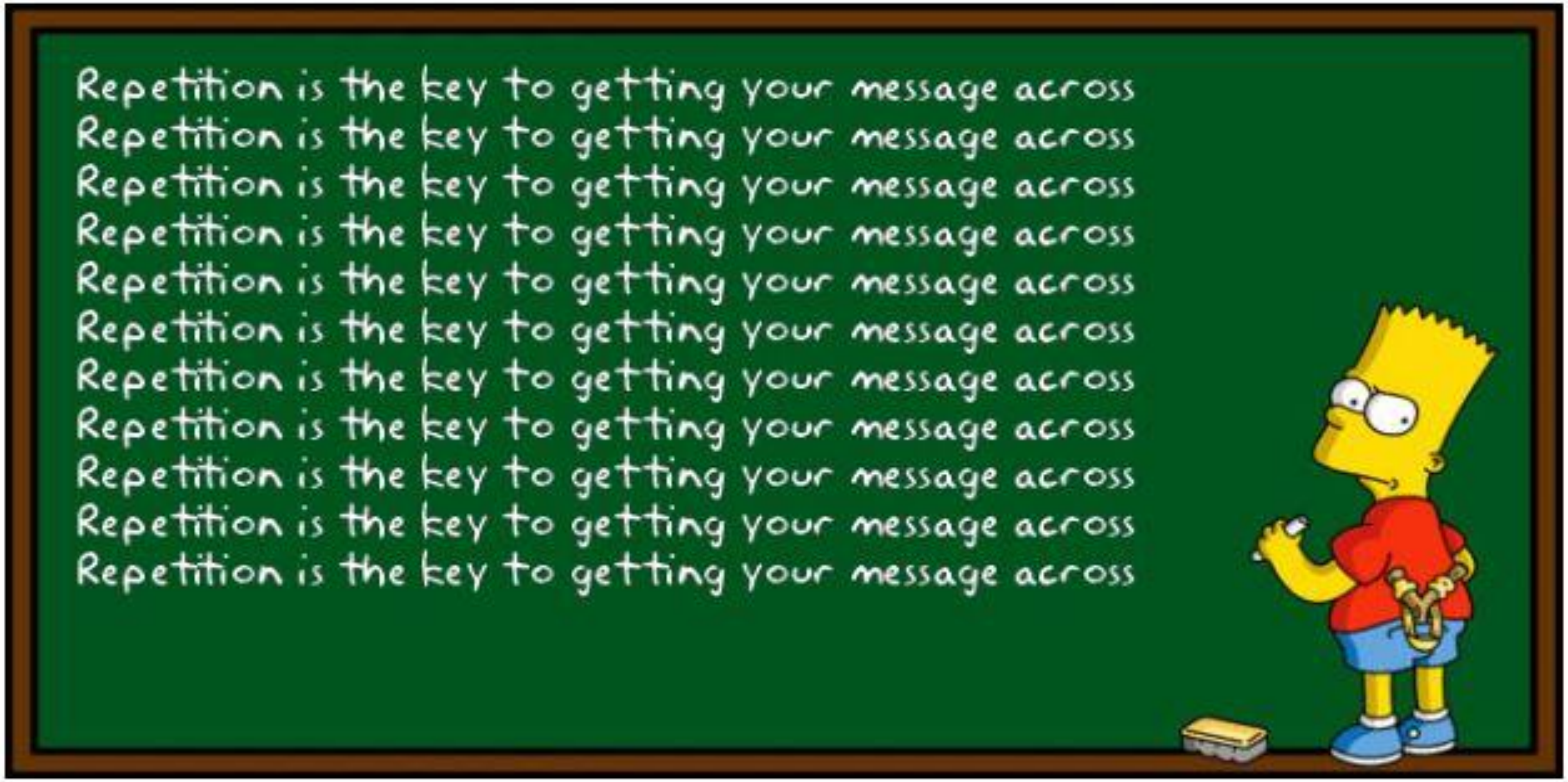


We can turn trajectories into sequences of letters
(Comparisons are a lot easier this way!)


Did the robot capture my intent?

Robust Robot Learning from Demonstration and Skill Repair Using Conceptual Constraints

[IROS 18]



Skills learned from demonstrations can be brittle due to the **limited information content** provided by trajectory demonstrations.

For example, a learned skill may only execute correctly for specific environment or object used during demonstration.

Learning implied constraints (e.g., cups need to be carried upright) from demonstrations can require a **prohibitively large** number of trajectories

Key Insights

Trajectory Demonstration

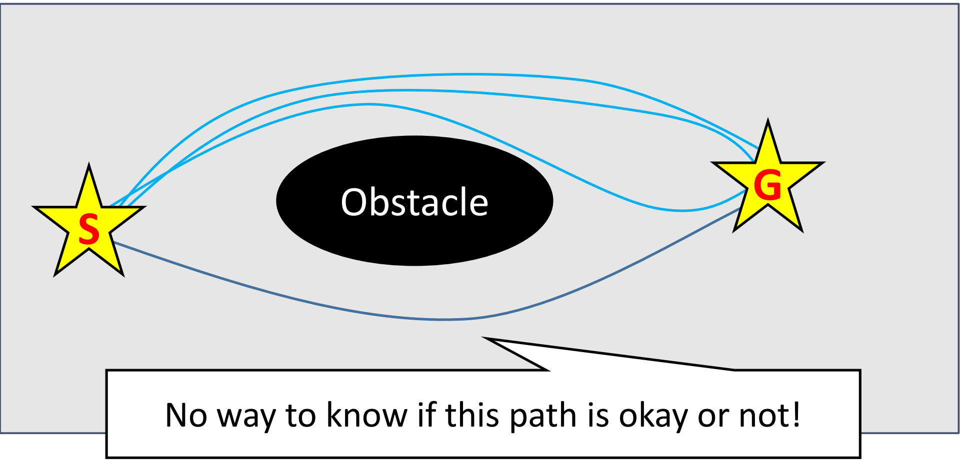
+ Intrinsically precise behavior specification

− Narrow coverage of skill per example

Narration

− Difficult to provide precise details

+ Can easily specify broadly applicable concepts



"Pick up the glass of water"
"Move it in an arc over the table to the bowl"
"But don't carry it over the laptop if it is full"
"Also make sure that your gripper stays closed"
"But not tight enough to break the glass"

…

Concept Constrained Learning from Demonstration

CC-LfD Algorithm
Augments Keyframe-based LfD by incorporating narrated high level constraints into keyframe models.

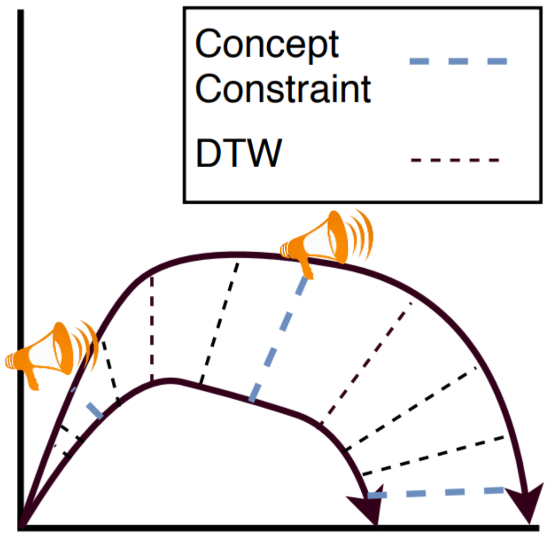
Conceptual Constraint
A physically grounded or abstract behavioral restriction encoded as a Boolean function

CC-LfD Allows You To:

Increase Skill Robustness	**Reduce Training Requirements**	**Increase Resilience to Poor Training**	**Improve and Repair Existing Skills**
Improves execution under conditions not seen during training	Learns more flexible, generalizable representations with less data	Avoids skill failures even when trained with sub-optimal demonstrations	Enables one-shot skill repair to improve existing skills with a single new example

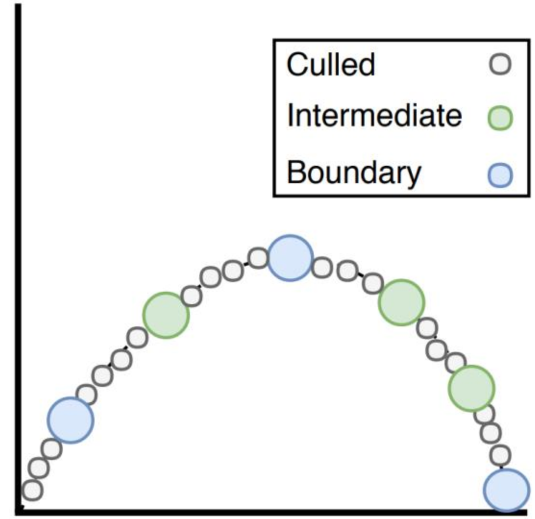
CC-LfD :: Algorithm Overview

1



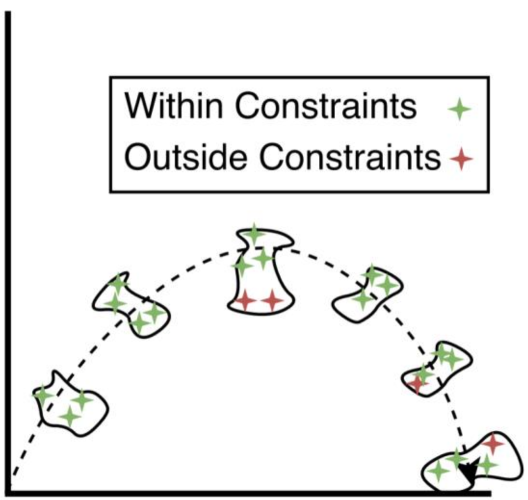
Record w/ Narration, & Align

2



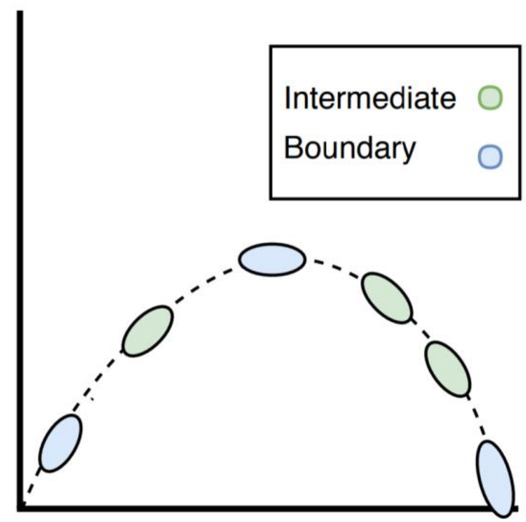
Cluster & Model Keyframes

3



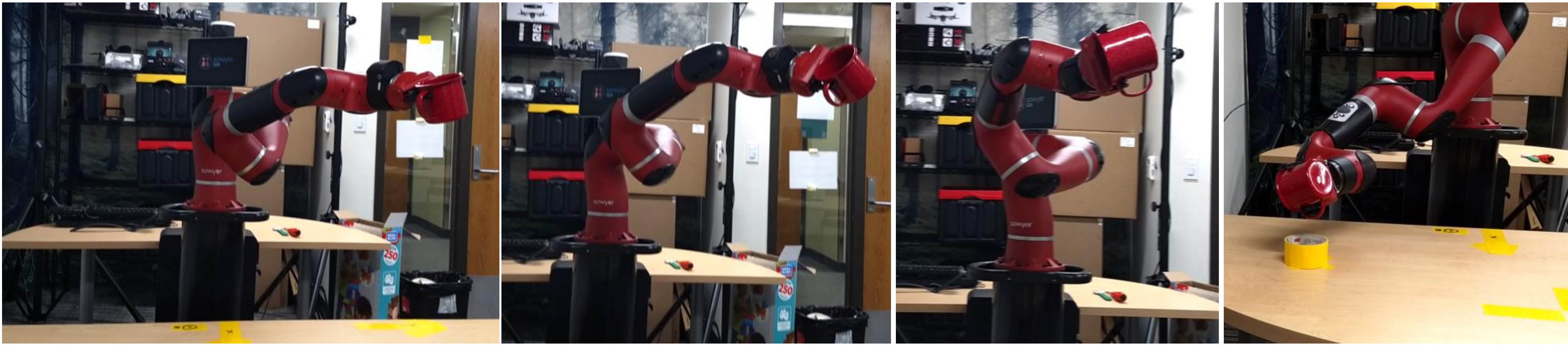
Rejection Sampling

4

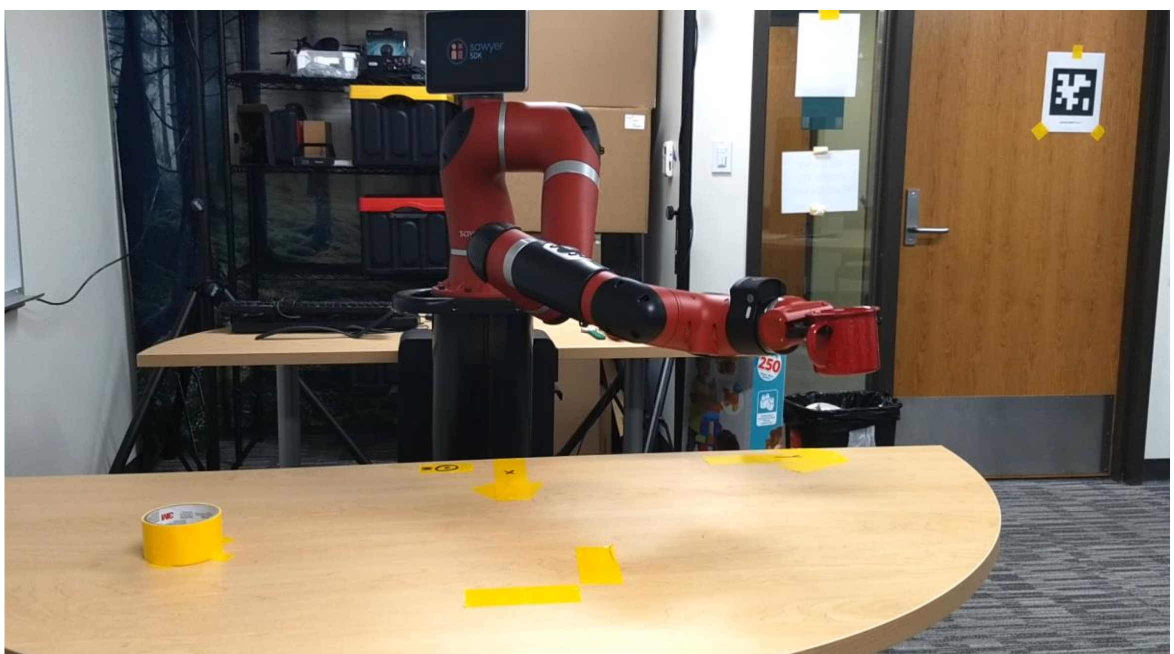


Remodel & Reconstruct

Unconstrained Skill Reconstruction from Keyframed Trajectories



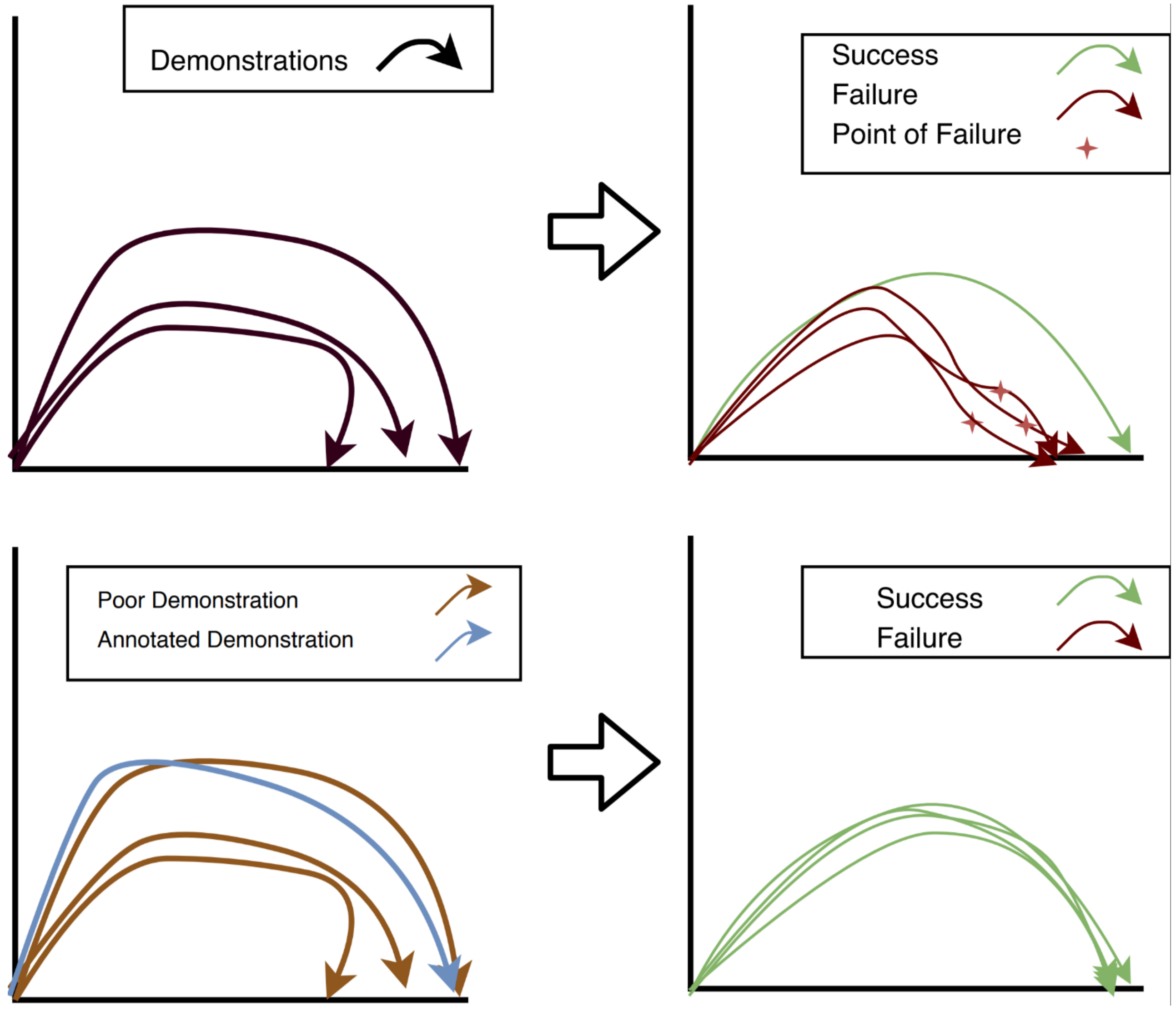
Skill Reconstruction from Keyframed Trajectories with CC-LfD Narration



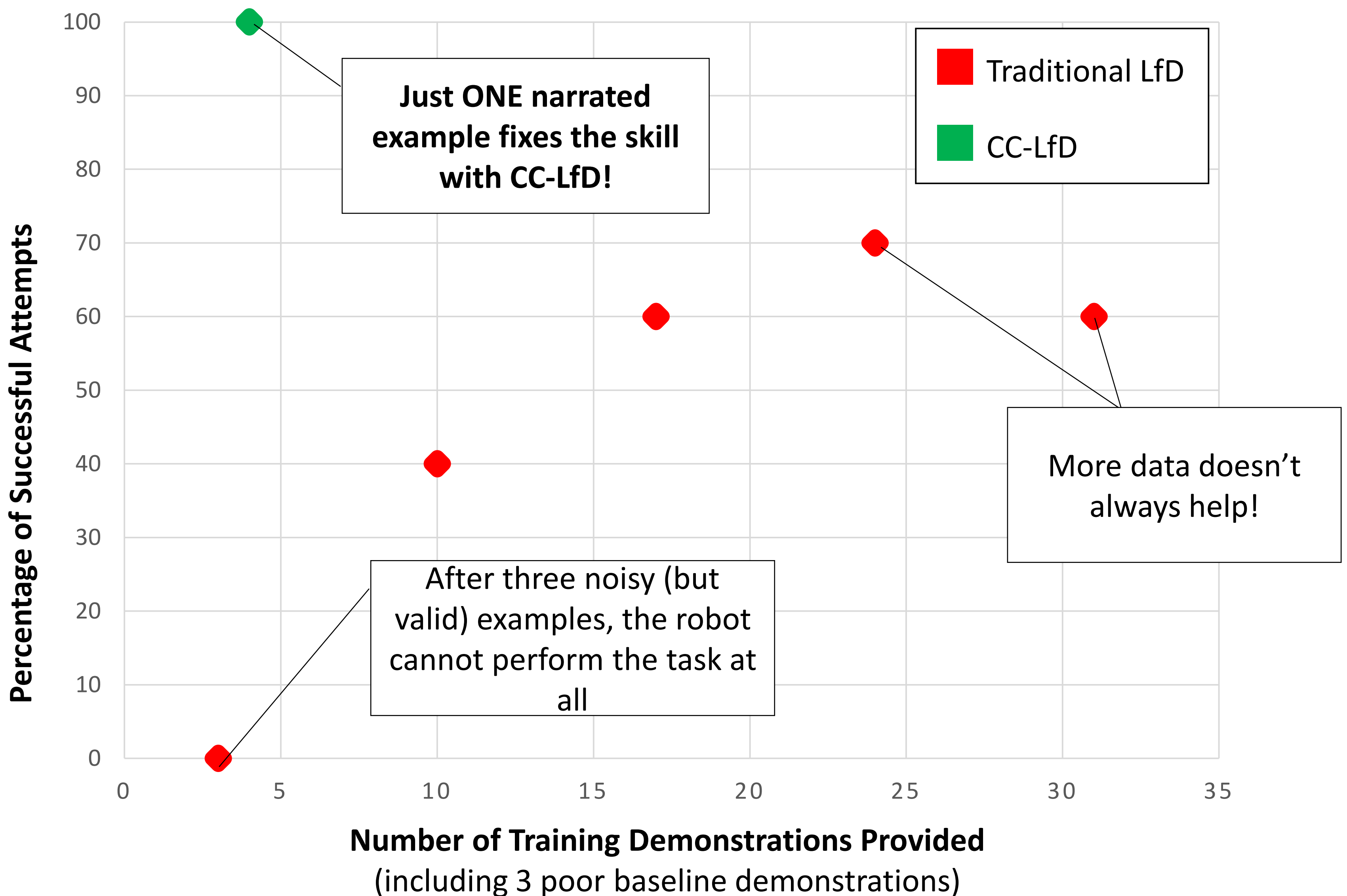
One-shot Skill Repair

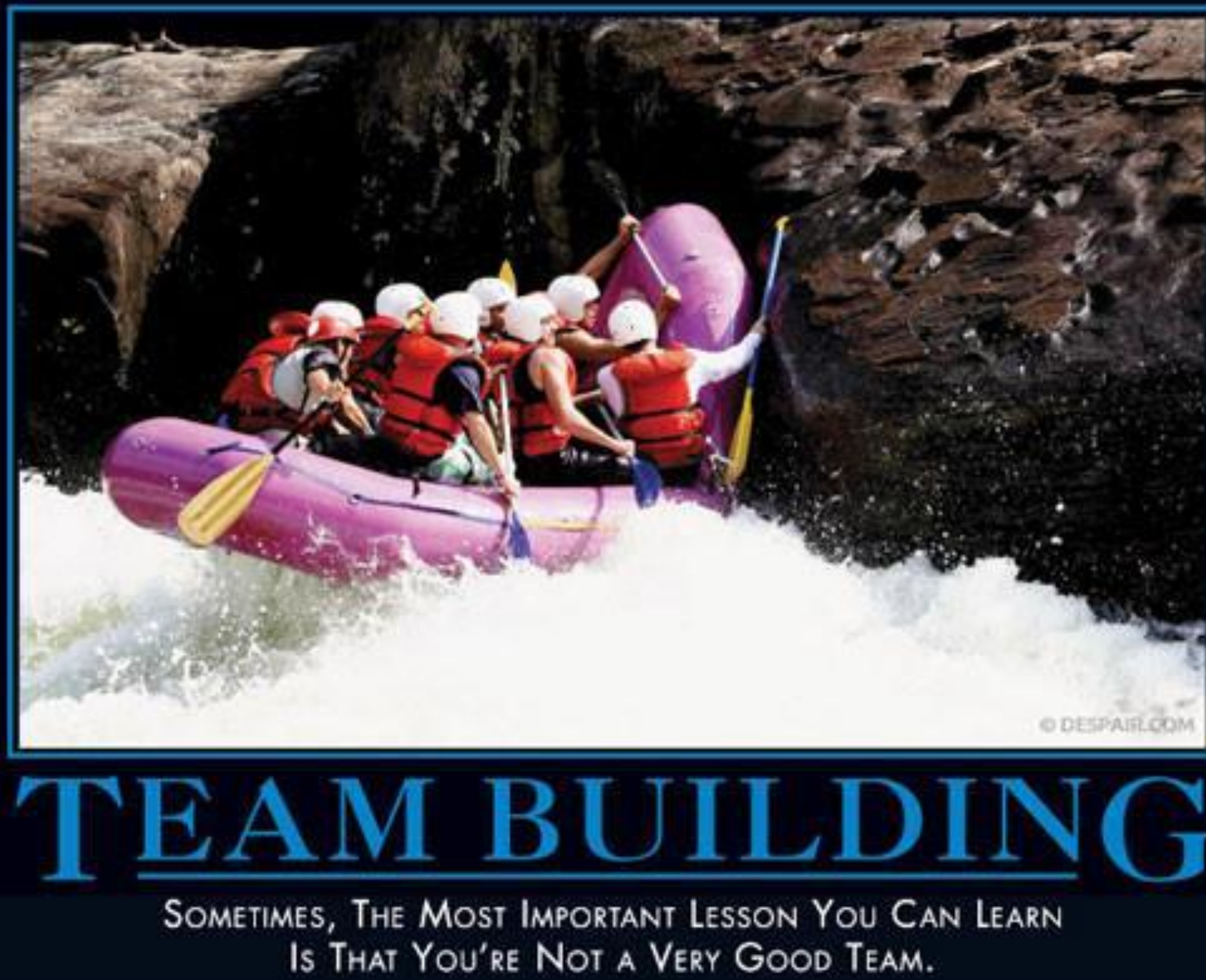
Broken Skill

Fixed Skill



“POURING TASK” ROBOT PERFORMANCE AND ONE-SHOT SKILL REPAIR




© DESPAIR.COM
TEAM BUILDING
SOMETIMES, THE MOST IMPORTANT LESSON YOU CAN LEARN
IS THAT YOU'RE NOT A VERY GOOD TEAM.

Teaming Paradigms

Leader / Follower

Equal Partners

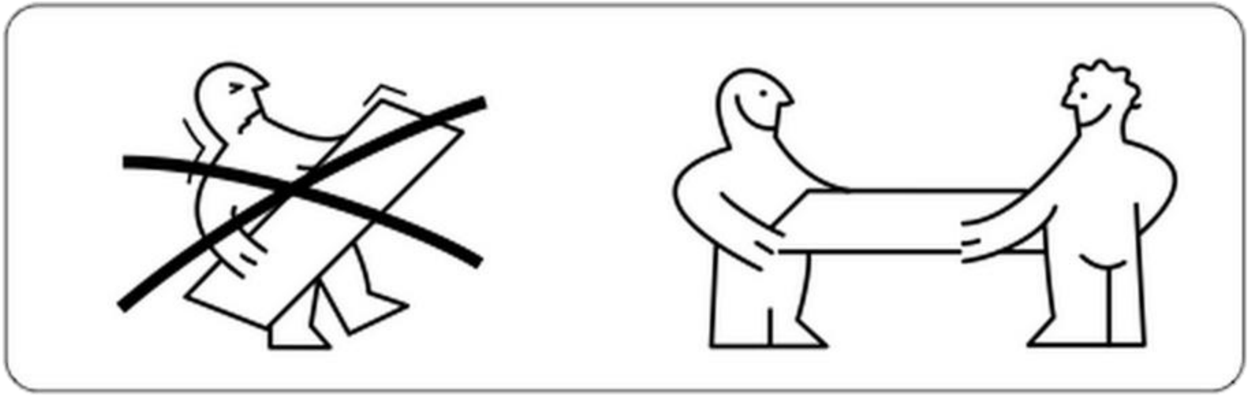
How can we enable collaborative robots that may

lack either *authority* or *capability*

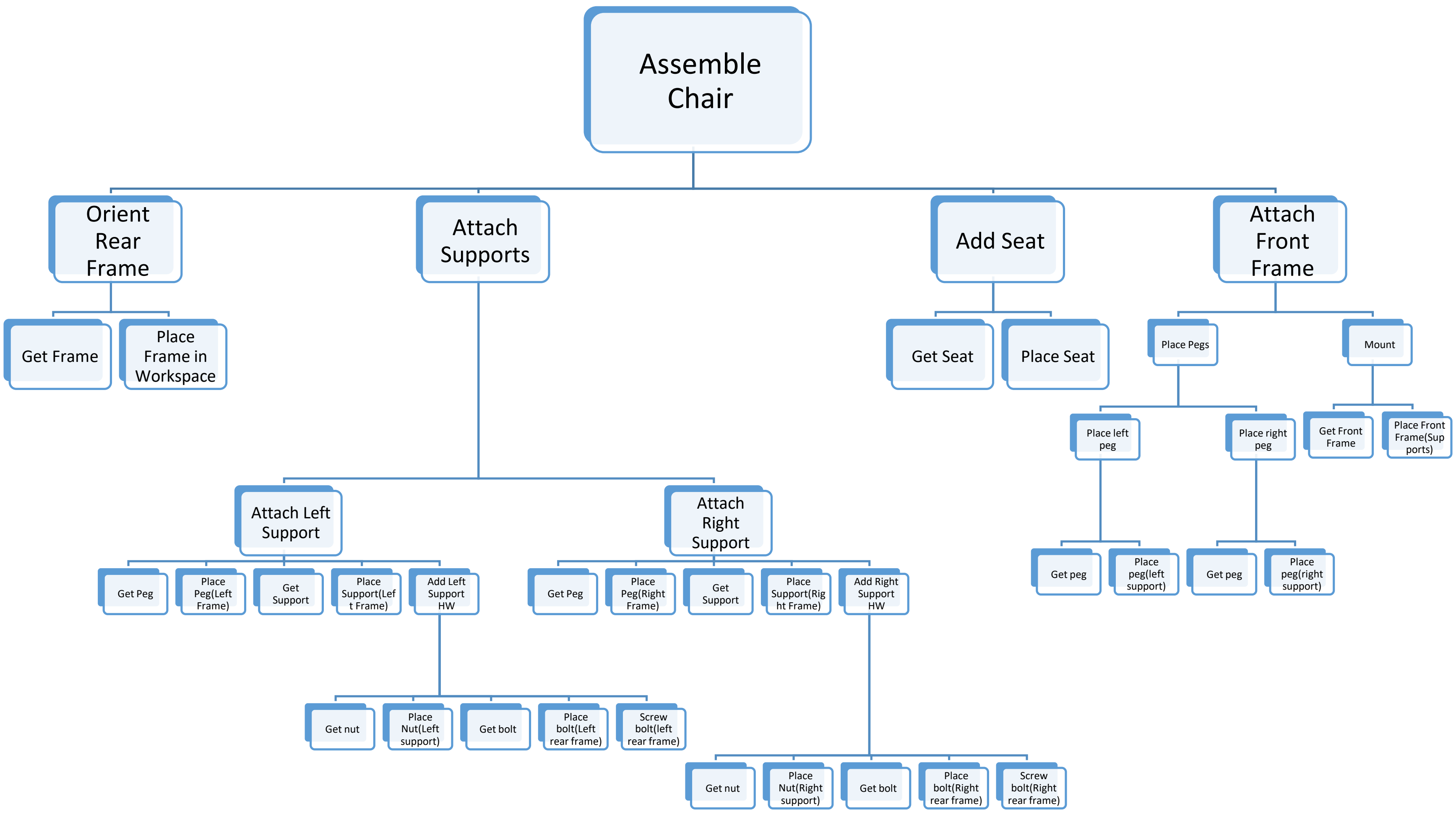
to provide utility to their co-workers?

Supportive Behaviors

Actions that facilitate more rapidly satisfiable or less difficult task solutions.



Hierarchical Task Structure
IKEA Chair



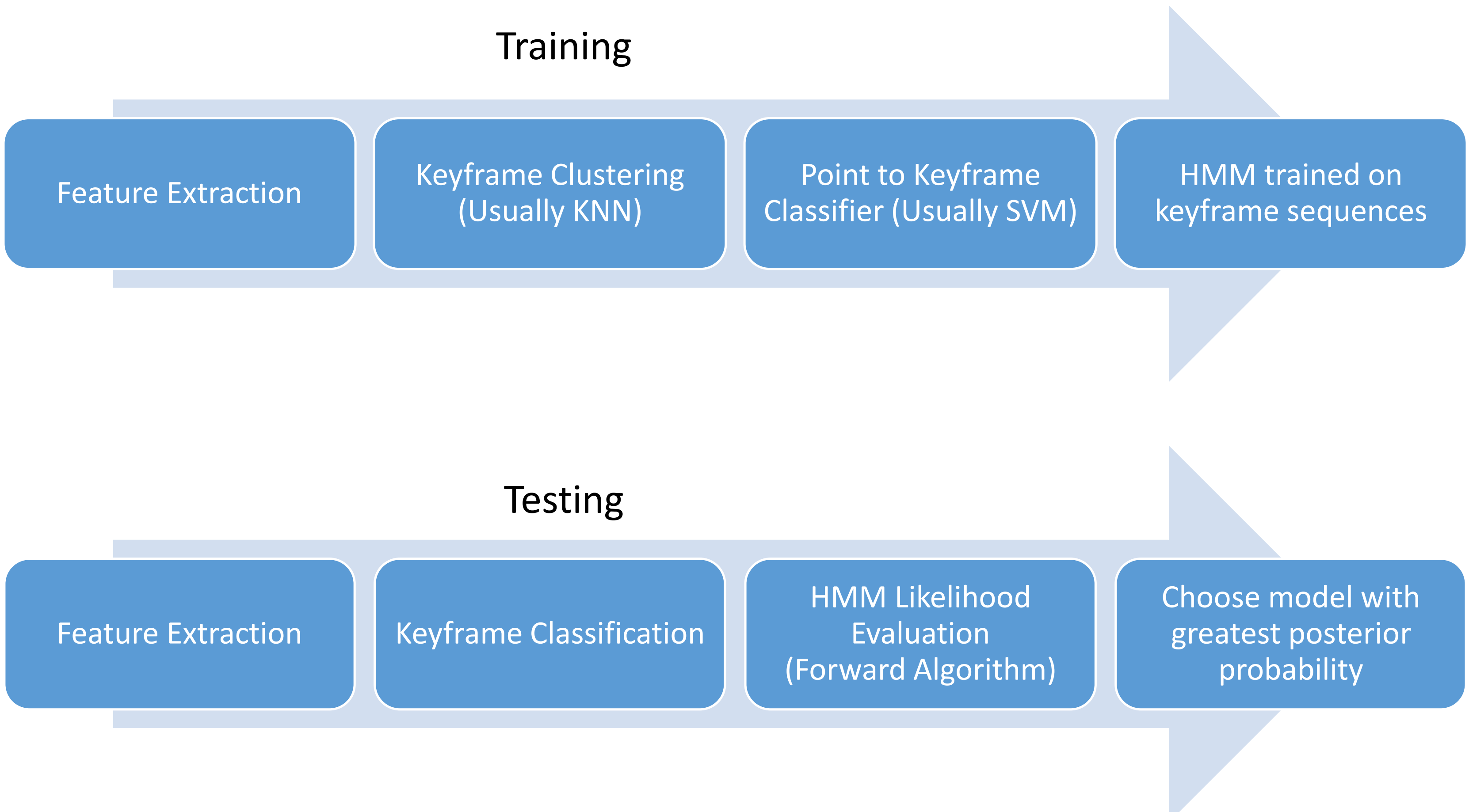
Collaborative robots need to recognize human activities

- Nearly all collaboration models depend on some form of activity recognition
- Collaboration imposes real-time constraints on classifier performance and tolerance to partial trajectories

Interpretable Models for Fast Activity Recognition and Anomaly Explanation During Collaborative Robotics Tasks

[ICRA 17]

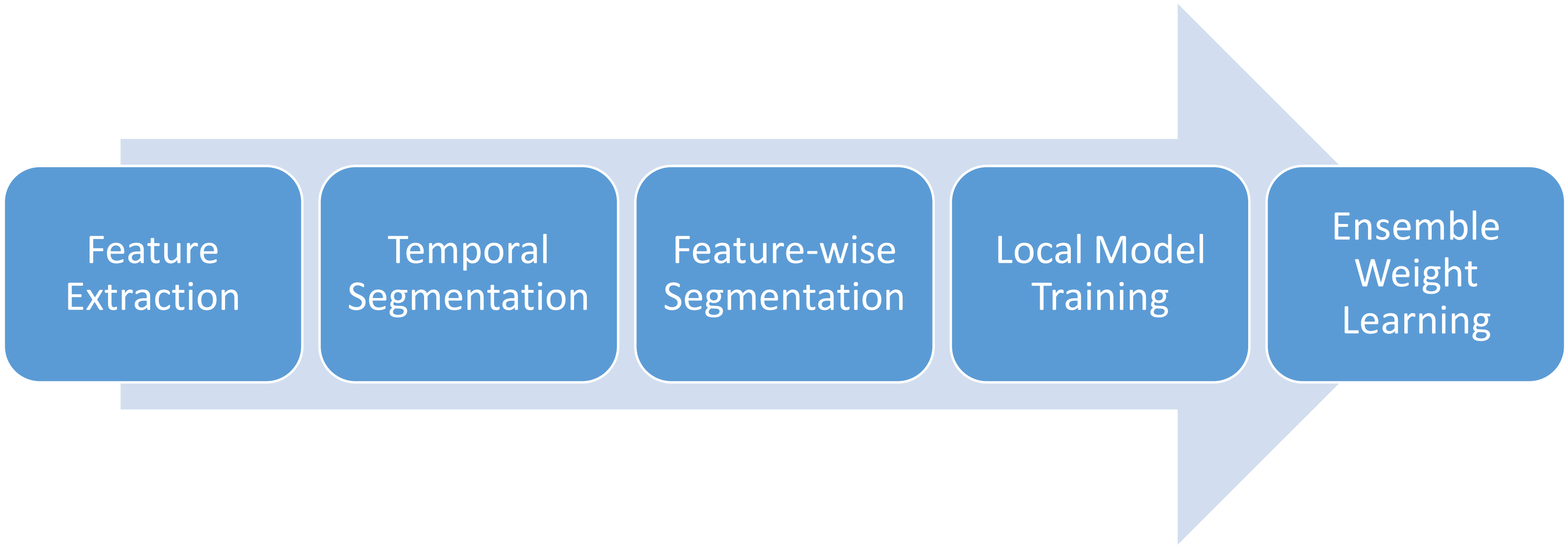
Common Activity Classifier Pipeline



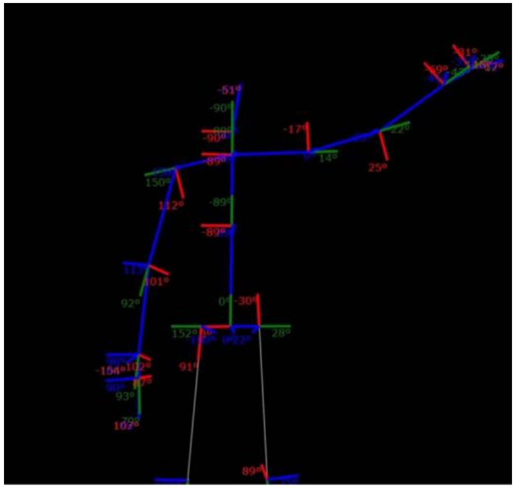
- P. Koniusz, A. Cherian, and F. Porikli, “Tensor representations via kernel linearization for action recognition from 3d skeletons.”
- Gori, J. Aggarwal, L. Matthies, and M. Ryoo, “Multitype activity recognition in robot-centric scenarios,”
- E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante, “A human activity recognition system using skeleton data from rgbd sensors.”
- L. Xia, C. Chen, and J. Aggarwal, “View invariant human action recognition using histograms of 3d joints.”

Rapid Activity Prediction Through Object-oriented Regression (RAPTOR)

A highly parallel ensemble classifier that is resilient to temporal variations

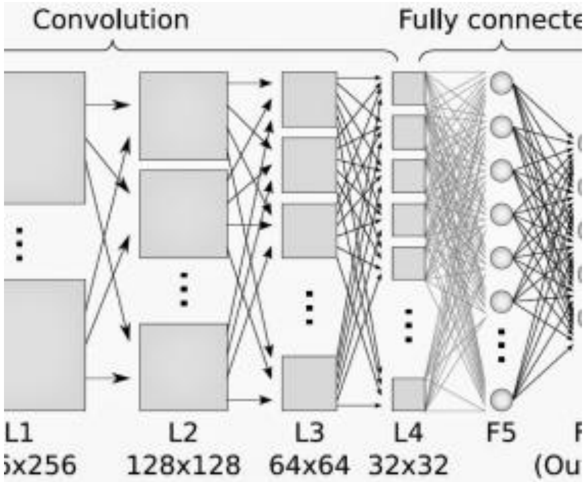


Activity Model Training Pipeline



Kinect Skeletal Joints

VICON Markers



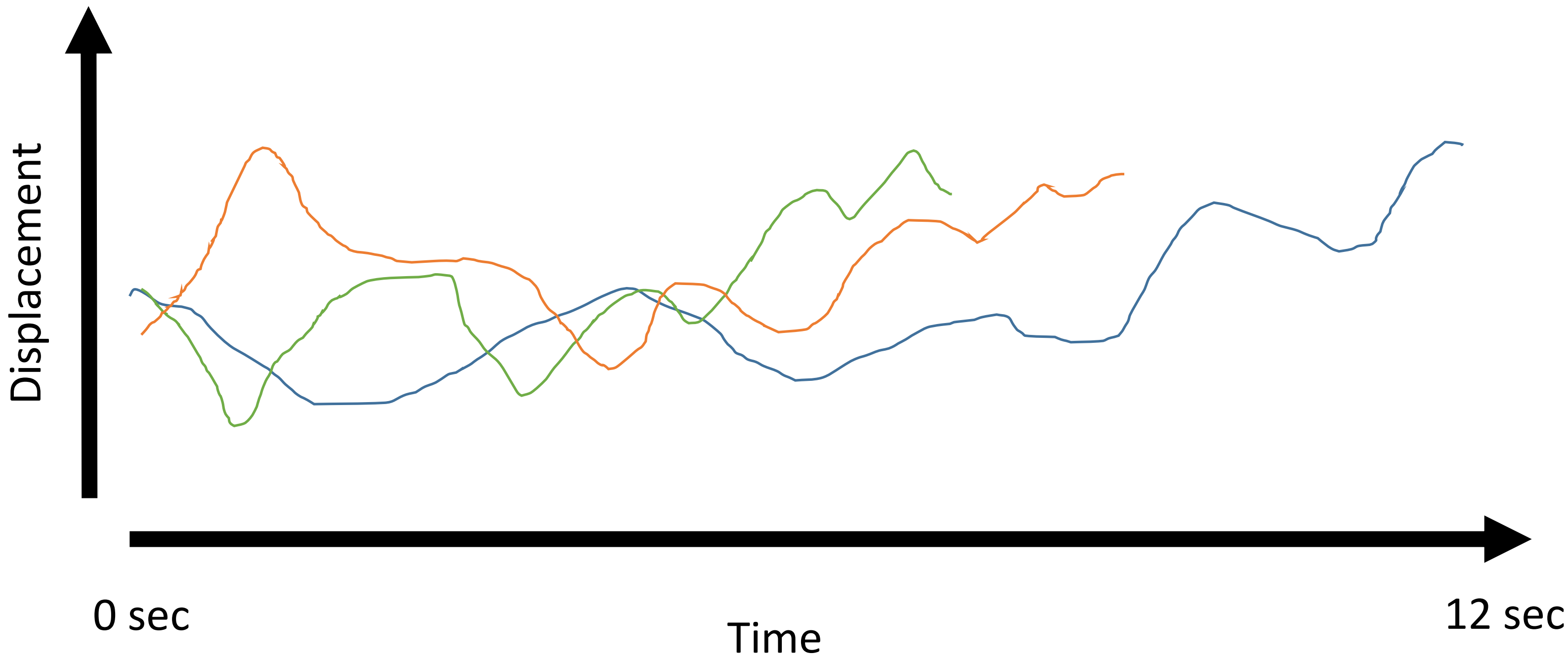
Learned Feature Extractor

$$\begin{pmatrix} 1. & 0.04 & -0.67 & -0.4 & -0.54 & -0.74 & -0.22 & -0.75 & -0.56 \\ 0.04 & 1. & 0.45 & 0.41 & -0.03 & -0.4 & -0.44 & -0.28 & 0.16 \\ -0.67 & 0.45 & 1. & 0.39 & 0.49 & 0.2 & -0.16 & 0.15 & 0.35 \\ -0.4 & 0.41 & 0.39 & 1. & 0.06 & 0.2 & 0.11 & 0.13 & 0.38 \\ -0.54 & -0.03 & 0.49 & 0.06 & 1. & 0.36 & 0.02 & 0.16 & 0.39 \\ -0.74 & -0.4 & 0.2 & 0.2 & 0.36 & 1. & 0.37 & 0.57 & 0.19 \\ -0.22 & -0.44 & -0.16 & 0.11 & 0.02 & 0.37 & 1. & 0.11 & -0.25 \\ -0.75 & -0.28 & 0.15 & 0.13 & 0.16 & 0.57 & 0.11 & 1 & 0.57 \end{pmatrix}$$

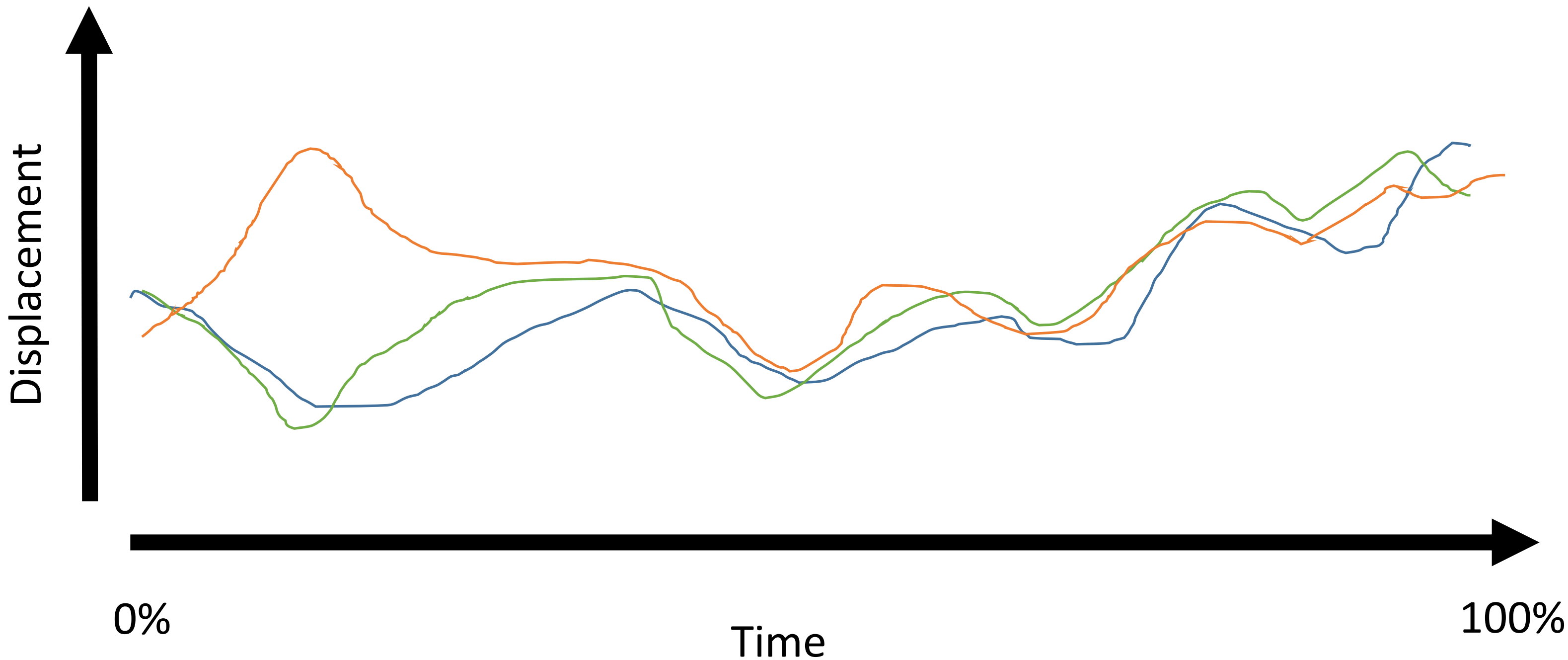
[Timestep x Feature] Matrix

Feature Extraction → Temporal Segmentation → Feature-wise Segmentation → Local Model Training → Ensemble Weight Learning

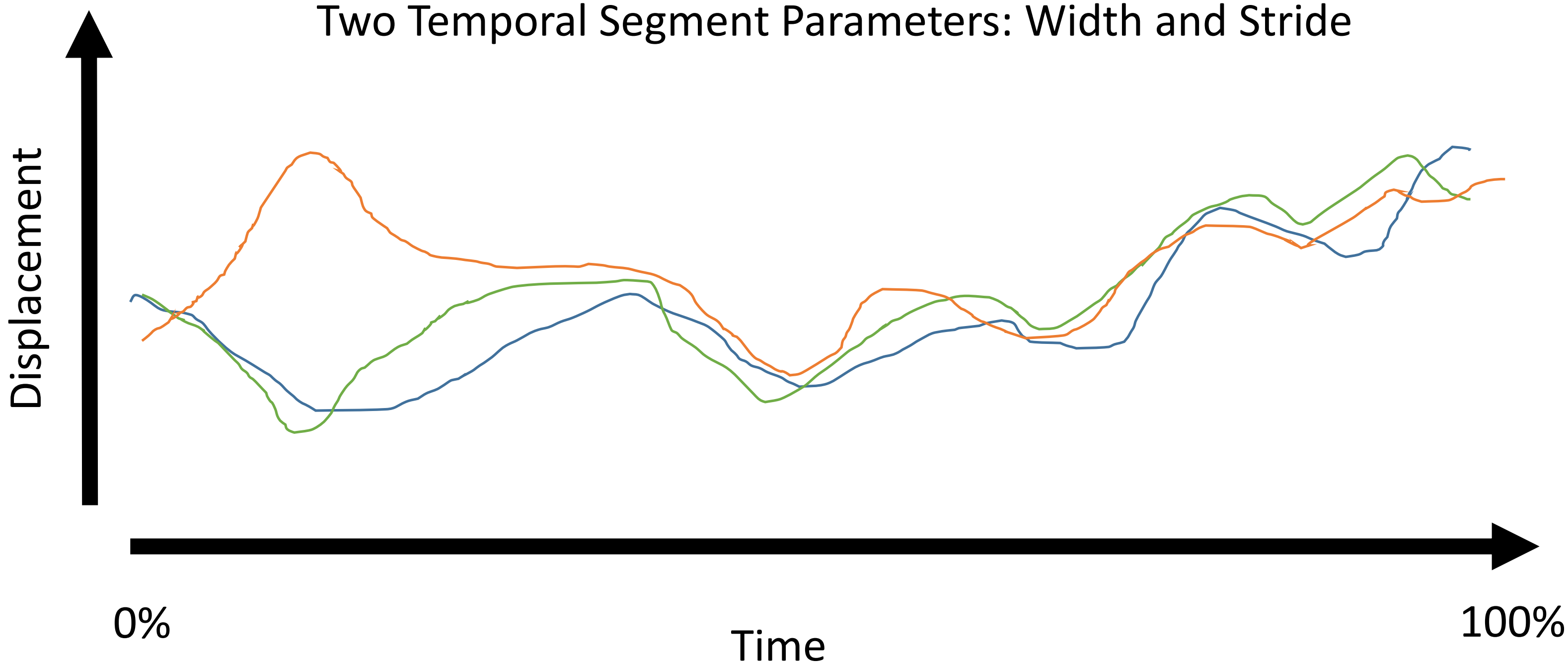
Activity Model Training Pipeline

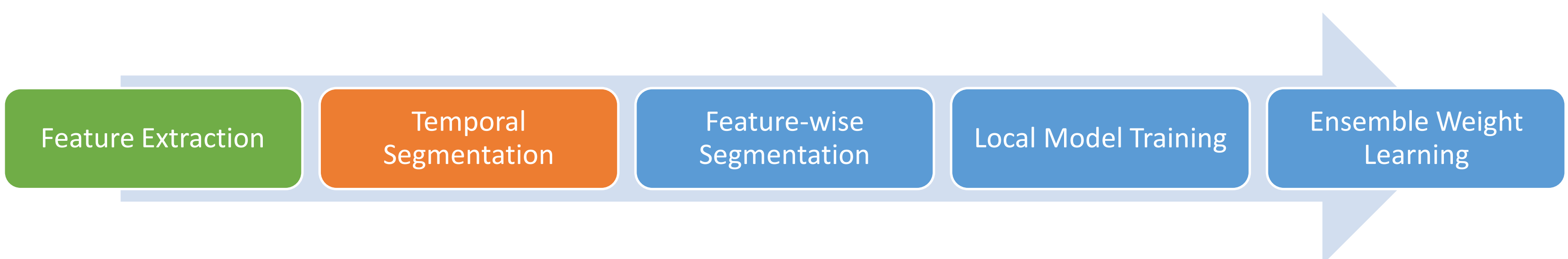


Activity Model Training Pipeline

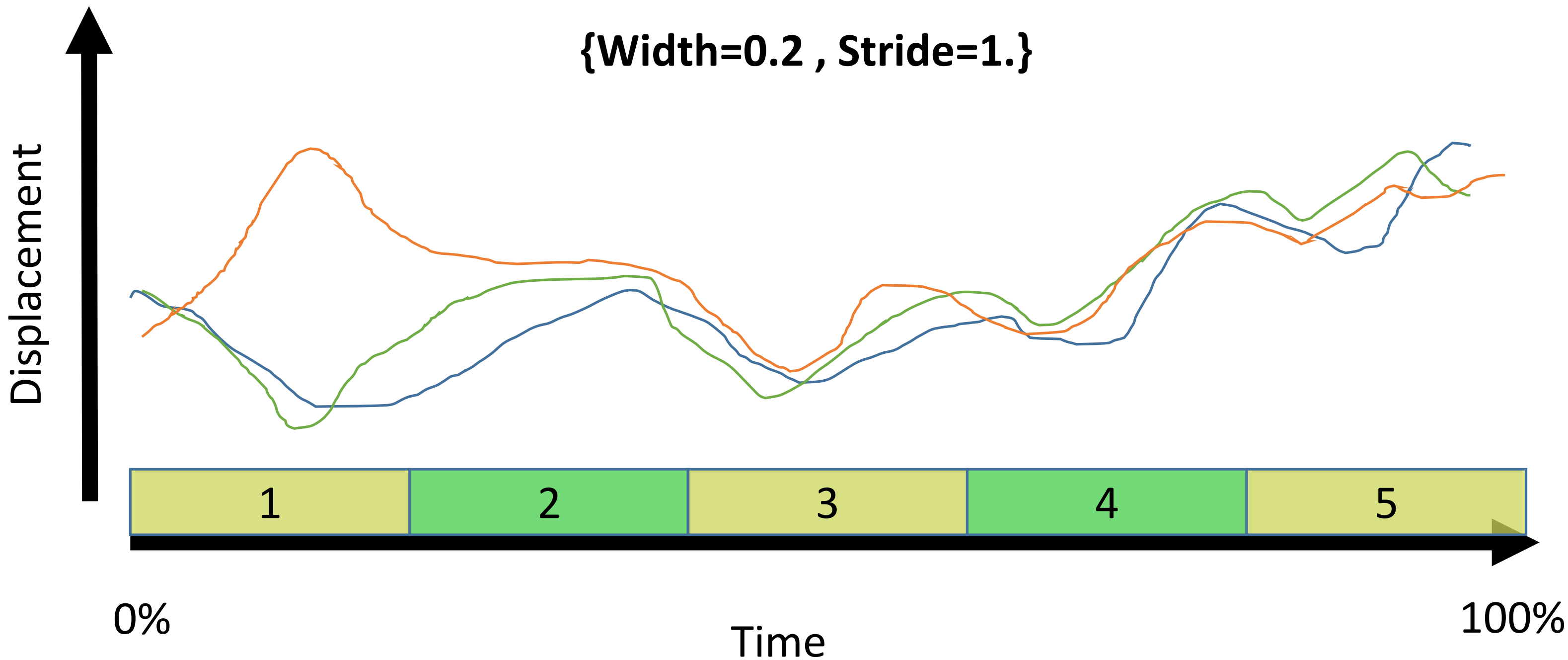


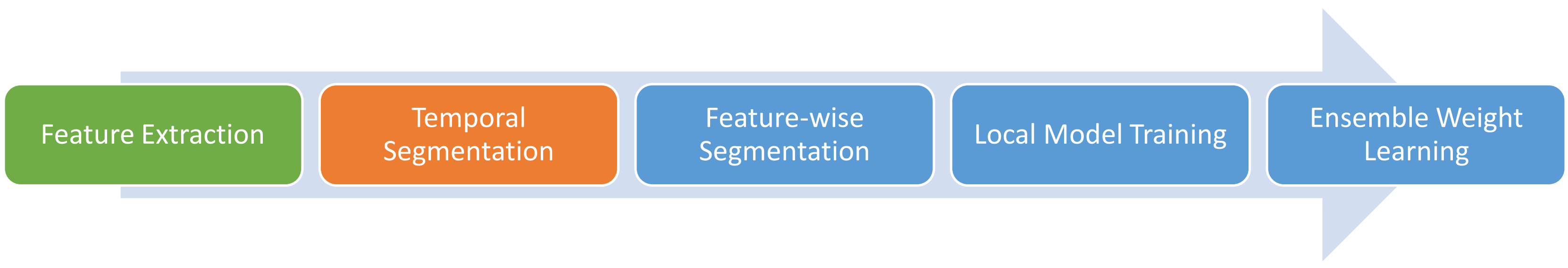
Activity Model Training Pipeline



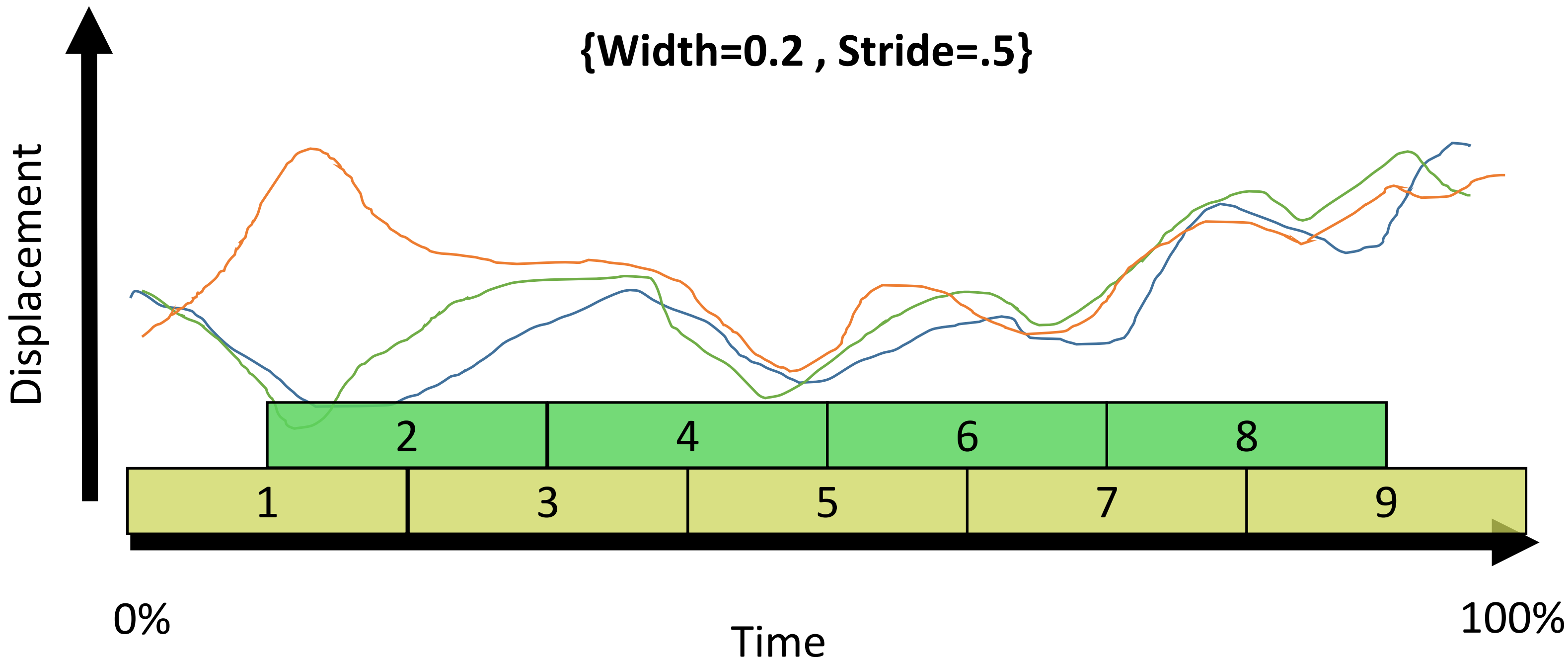


Activity Model Training Pipeline

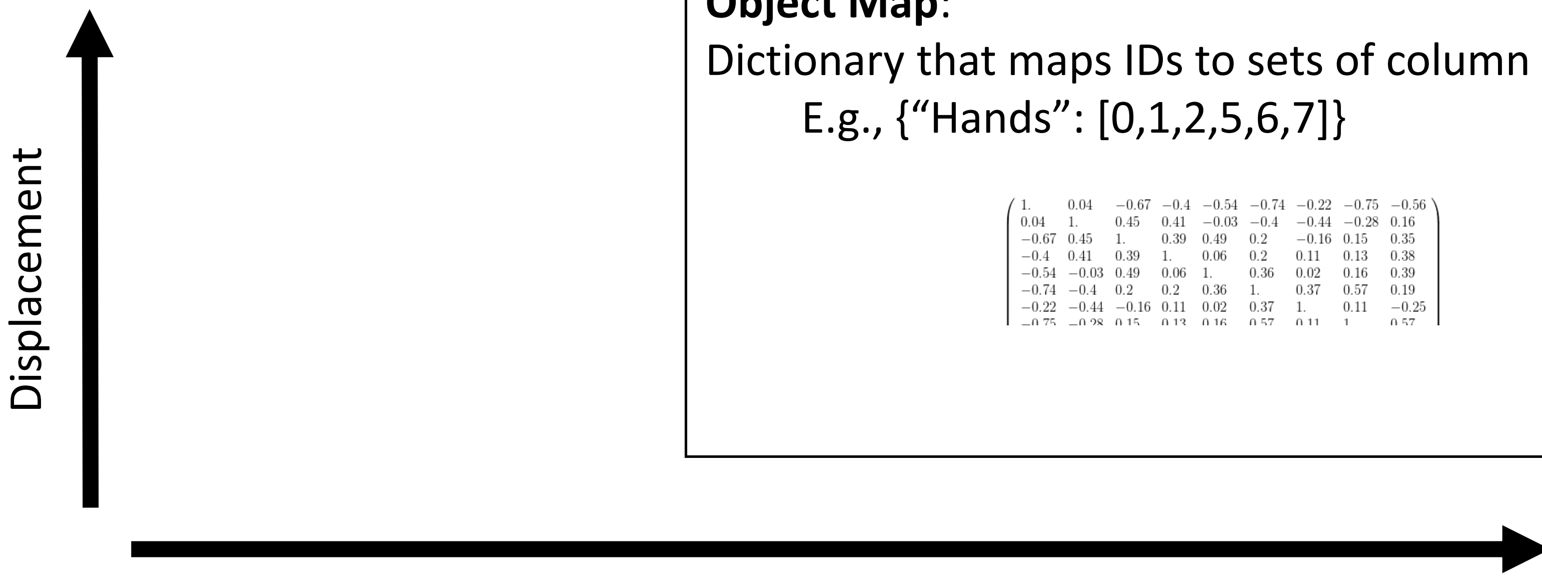




Activity Model Training Pipeline



Activity Model Training Pipeline

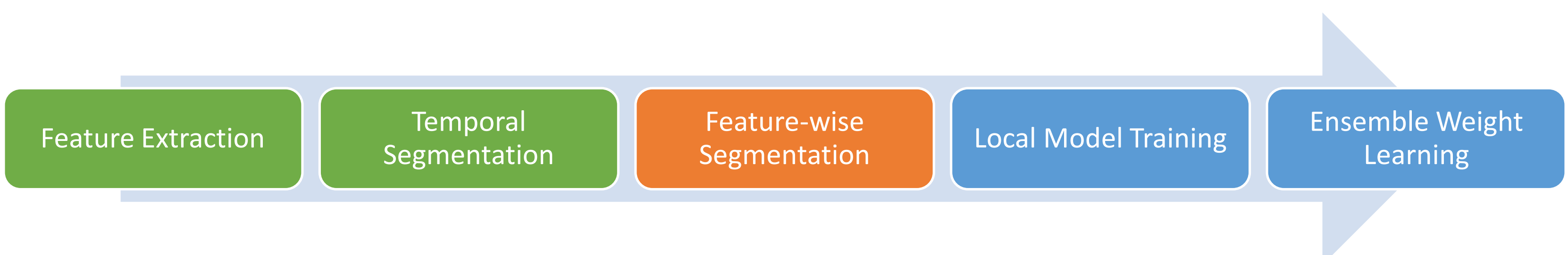


Object Map:

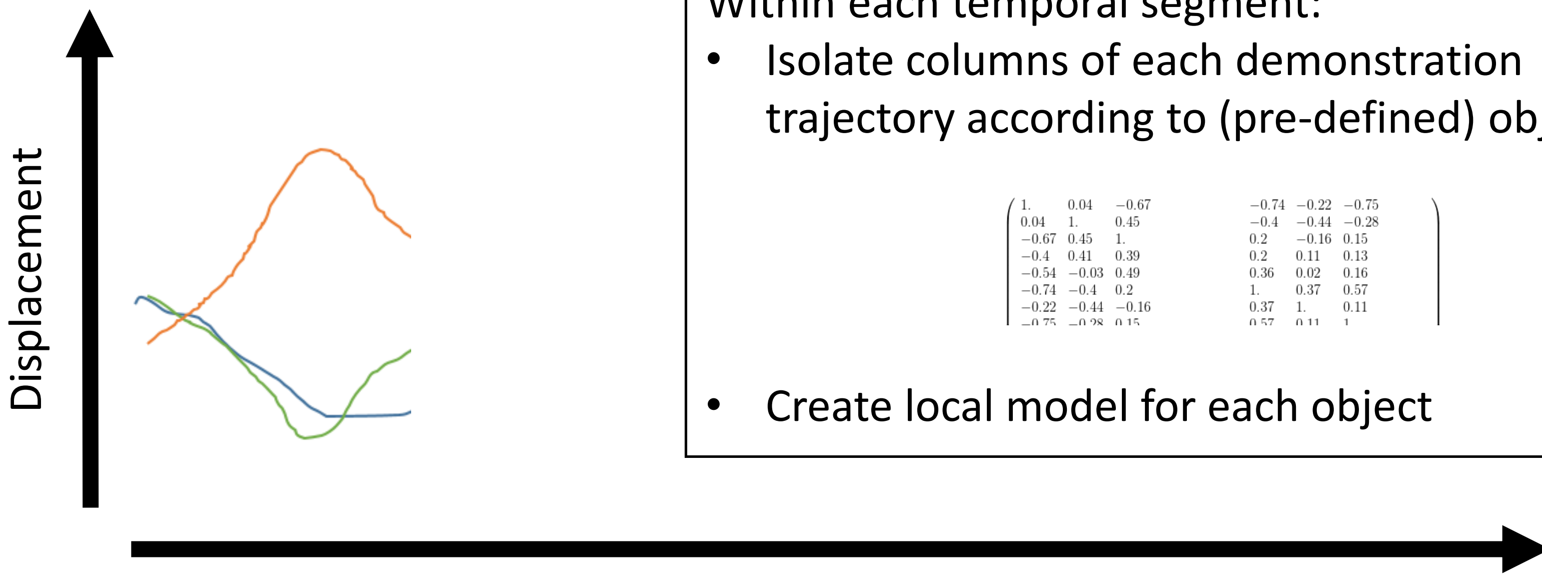
Dictionary that maps IDs to sets of column indices

E.g., {“Hands”: [0,1,2,5,6,7]}

$$\begin{pmatrix}
1. & 0.04 & -0.67 & -0.4 & -0.54 & -0.74 & -0.22 & -0.75 & -0.56 \\
0.04 & 1. & 0.45 & 0.41 & -0.03 & -0.4 & -0.44 & -0.28 & 0.16 \\
-0.67 & 0.45 & 1. & 0.39 & 0.49 & 0.2 & -0.16 & 0.15 & 0.35 \\
-0.4 & 0.41 & 0.39 & 1. & 0.06 & 0.2 & 0.11 & 0.13 & 0.38 \\
-0.54 & -0.03 & 0.49 & 0.06 & 1. & 0.36 & 0.02 & 0.16 & 0.39 \\
-0.74 & -0.4 & 0.2 & 0.2 & 0.36 & 1. & 0.37 & 0.57 & 0.19 \\
-0.22 & -0.44 & -0.16 & 0.11 & 0.02 & 0.37 & 1. & 0.11 & -0.25 \\
-0.75 & -0.28 & 0.15 & 0.13 & 0.16 & 0.57 & 0.11 & 1. & 0.57
\end{pmatrix}$$



Activity Model Training Pipeline

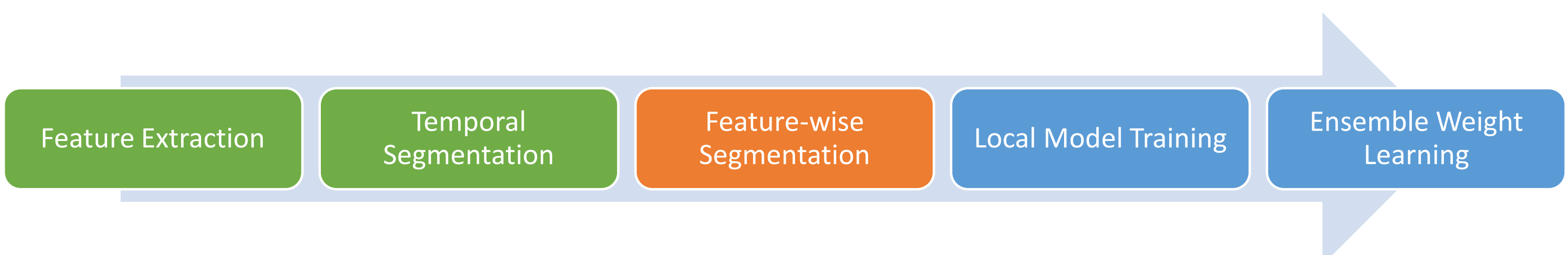


Within each temporal segment:

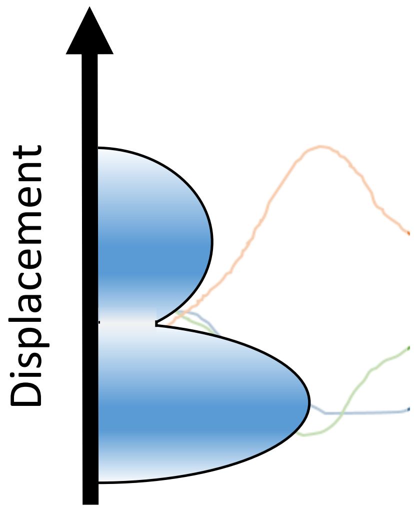
- Isolate columns of each demonstration trajectory according to (pre-defined) object map

$$\begin{pmatrix} 1. & 0.04 & -0.67 & \quad & -0.74 & -0.22 & -0.75 \\ 0.04 & 1. & 0.45 & & -0.4 & -0.44 & -0.28 \\ -0.67 & 0.45 & 1. & & 0.2 & -0.16 & 0.15 \\ -0.4 & 0.41 & 0.39 & & 0.2 & 0.11 & 0.13 \\ -0.54 & -0.03 & 0.49 & & 0.36 & 0.02 & 0.16 \\ -0.74 & -0.4 & 0.2 & & 1. & 0.37 & 0.57 \\ -0.22 & -0.44 & -0.16 & & 0.37 & 1. & 0.11 \\ -0.75 & -0.28 & 0.15 & & 0.57 & 0.11 & 1. \end{pmatrix}$$

- Create local model for each object



Activity Model Training Pipeline



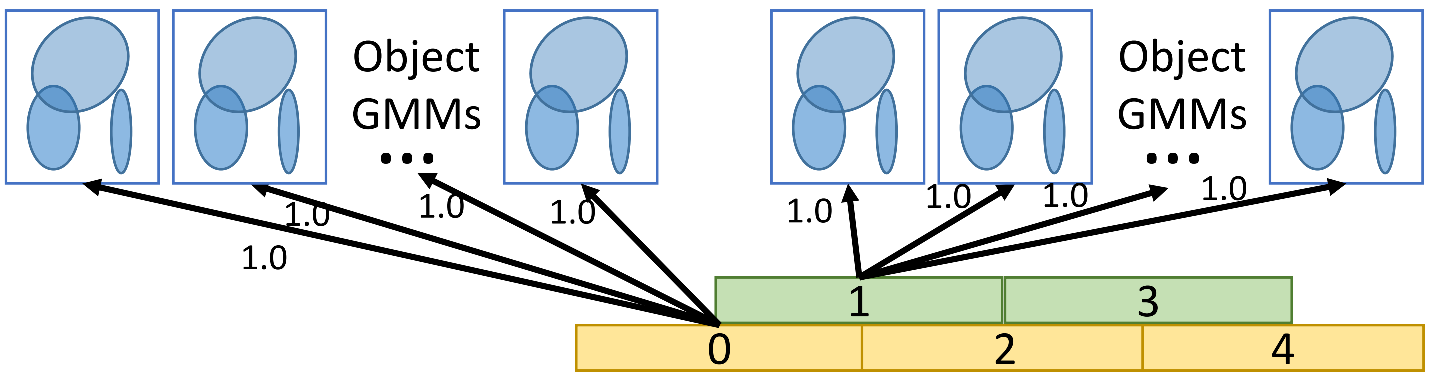
Within each temporal-object segment:

- Ignore temporal information for each data point
- Treat as general pattern recognition problem
- **Model the resulting distribution using a GMM**

Result: An activity classifier ensemble across objects and time!

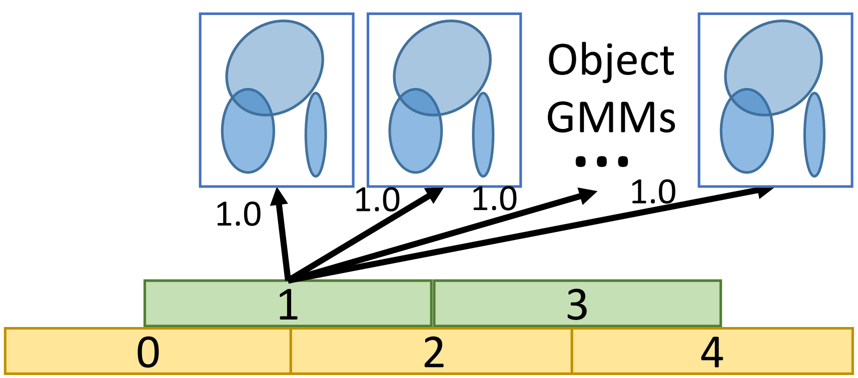
Activity Model Training Pipeline

Need to find the most discriminative Object GMMs per time segment

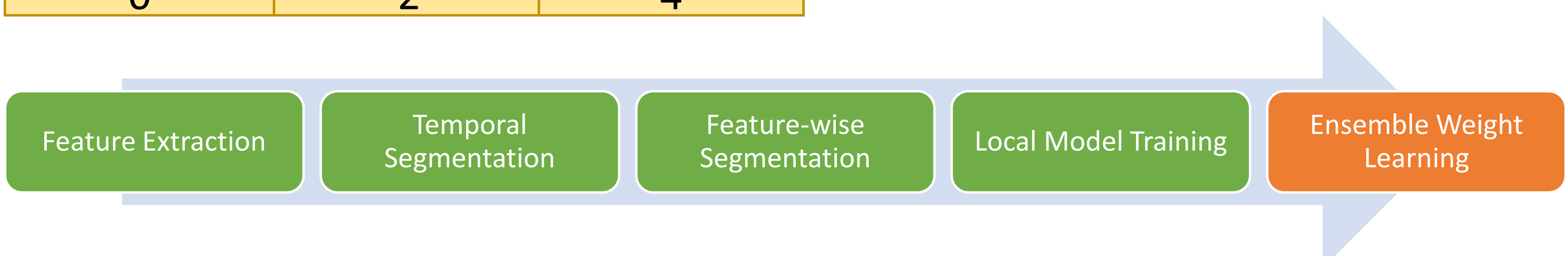


Activity Model Training Pipeline

Need to find the most discriminative Object GMMs per time segment

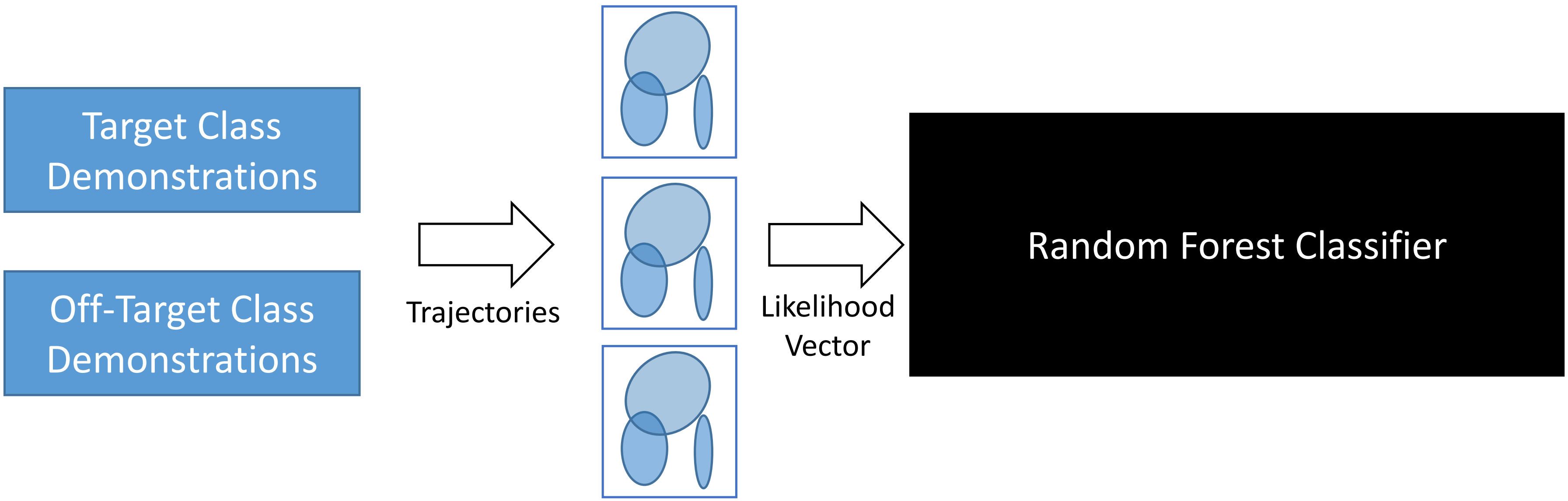


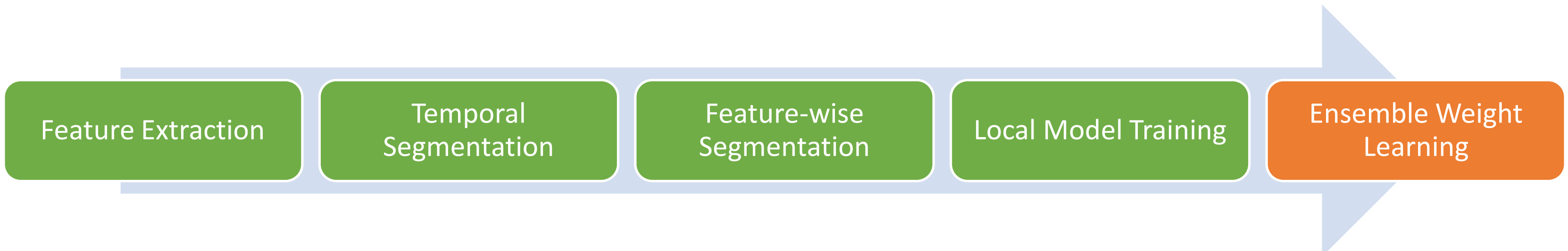
Random Forest Classifier



Activity Model Training Pipeline

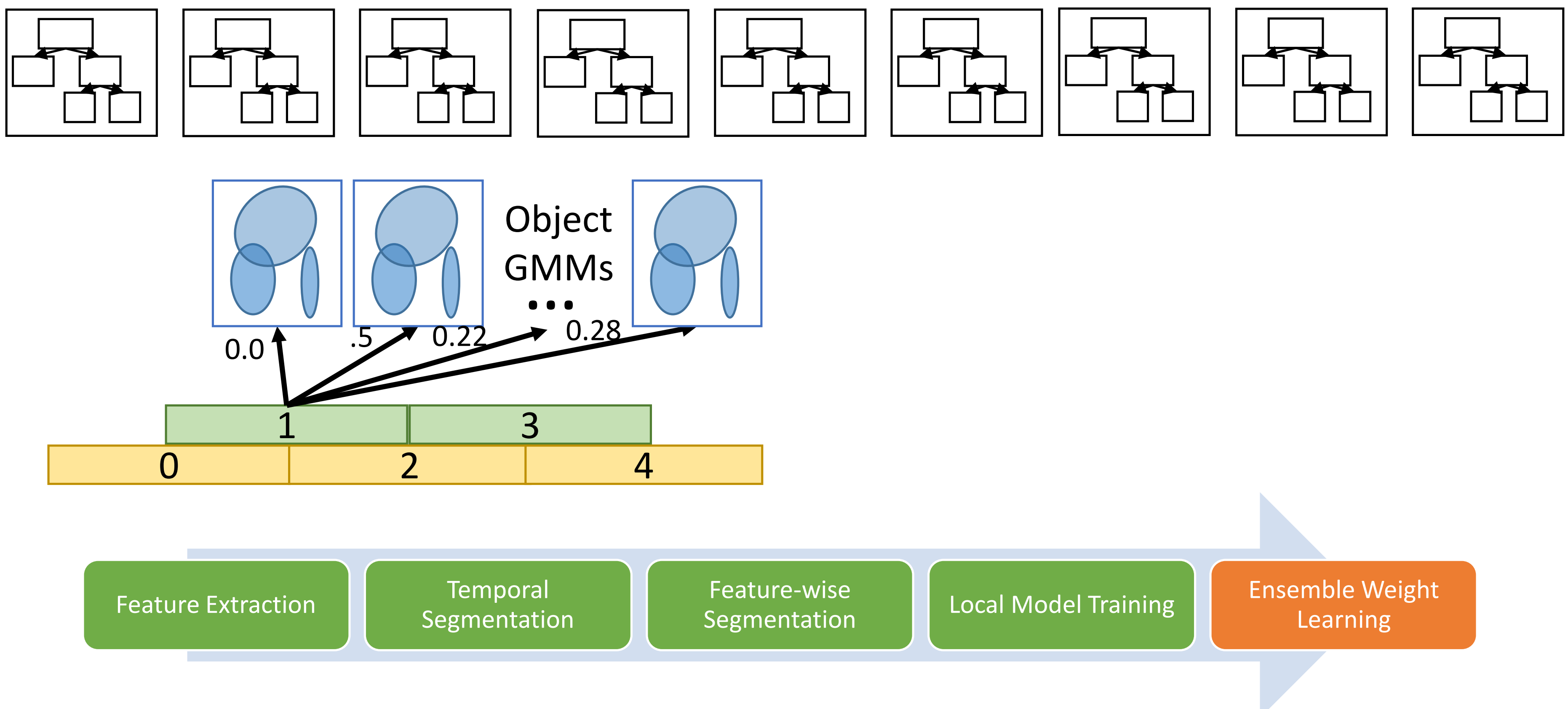
Need to find the most discriminative Object GMMs per time segment





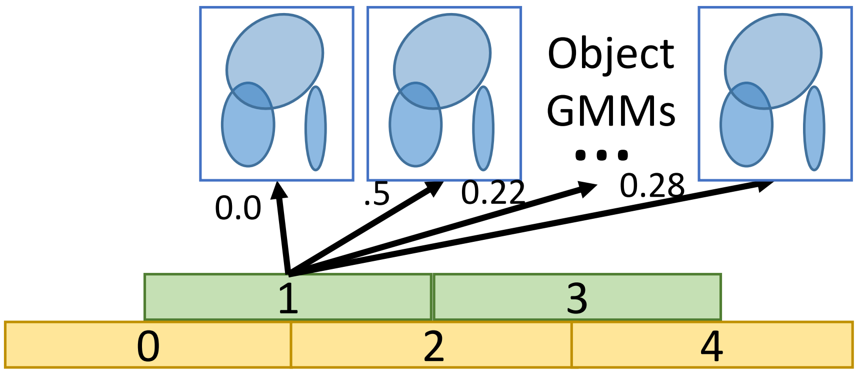
Activity Model Training Pipeline

- Choose top-N most discriminative features from the Random Forest classifier
- Weight each GMM proportional to its discriminative power



Activity Model Training Pipeline

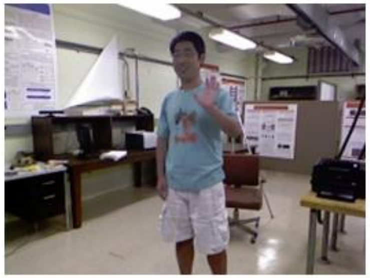
- Choose top-N most discriminative object-based classifiers
- Weight each object proportionally to its discriminative power



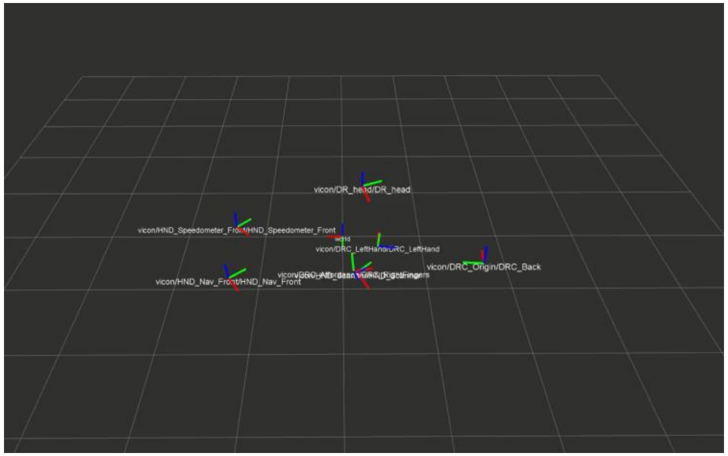
Result: Trained Highly Parallel Ensemble Learner with Temporal/Object-specific sensitivity

Results: Three Datasets

- **UTKinect** publicly available benchmark (Kinect Joints)
- **Dynamic** Actor Industrial Manufacturing Task (Joint positions)
- **Static** Actor Industrial Manufacturing Task (Joint positions)



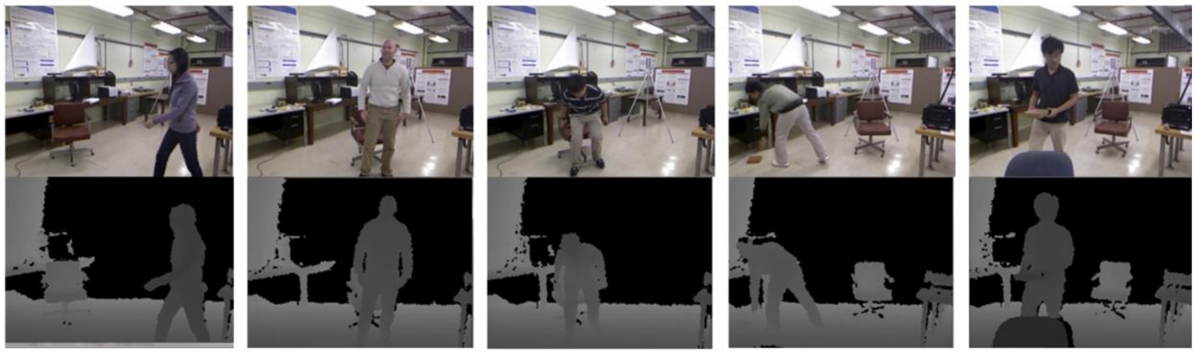
UTKinect



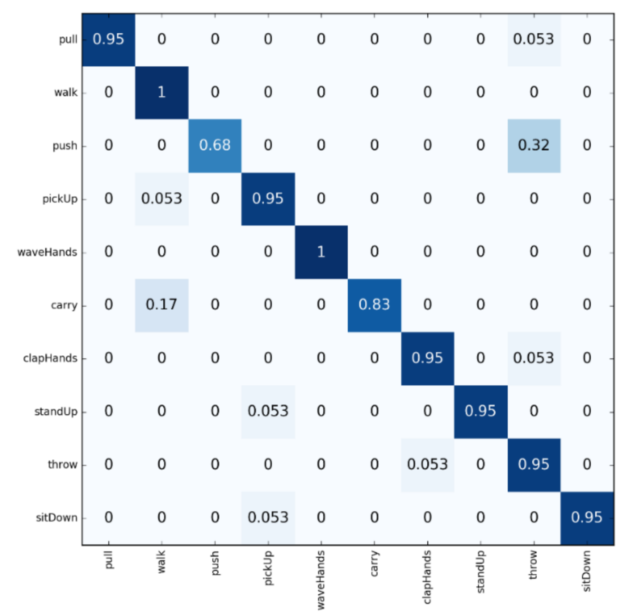
Automotive Final Assembly

Sealant Application

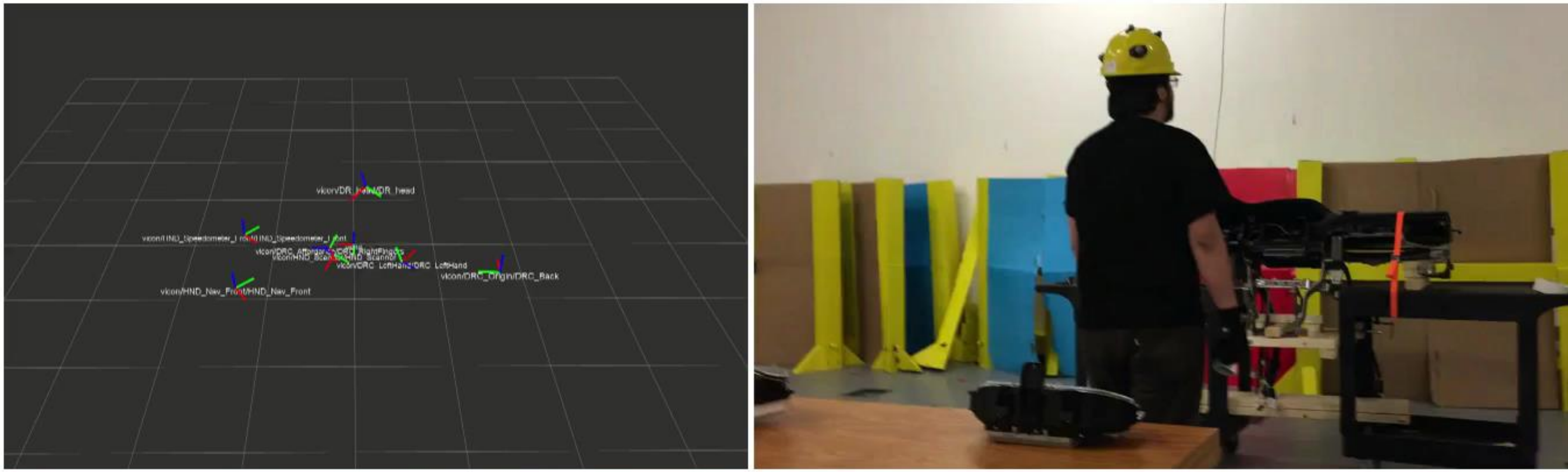
Recognition Results: UTKinect-Action3D



Real-time UTKinect Activity Recognition Accuracy	
Classifier	**Accuracy**
Slama et al. (2015) [21]	88.5%
Chrungoo et al. (2014) [18]	89.45%
Xia et al. (2012) [11]	90.9%
Wang et al. (2015) [24]	90.9%
Devanne et al. (2013) [20]	91.5%
RAPTOR (proposed method)	**92.1%**



Results: Online Prediction



```
Elapsed Time: 0.1sec     Classified activity move_to_dash with likelihood 0.84128       Ground Truth: None
Elapsed Time: 0.13sec    Classified activity move_to_dash with likelihood 0.84811       Ground Truth: None
Elapsed Time: 0.17sec    Classified activity move_to_dash with likelihood 0.86419       Ground Truth: None
Elapsed Time: 0.2sec     Classified activity move_to_dash with likelihood 0.867         Ground Truth: None
Elapsed Time: 0.23sec    Classified activity move_to_dash with likelihood 0.95099       Ground Truth: None
```

RAPTOR Online Activity Prediction Accuracy				
Dataset	**25%**	**50%**	**75%**	**100%**
UTKinect	79.4%	83.1%	84.7%	92.1%
Static-Reach	69.7%	77.2%	93.8%	97.5%
Dynamic-AutoFA	91.7%	88.1%	90.5%	92.0%

Interpretability: Explaining Classifications

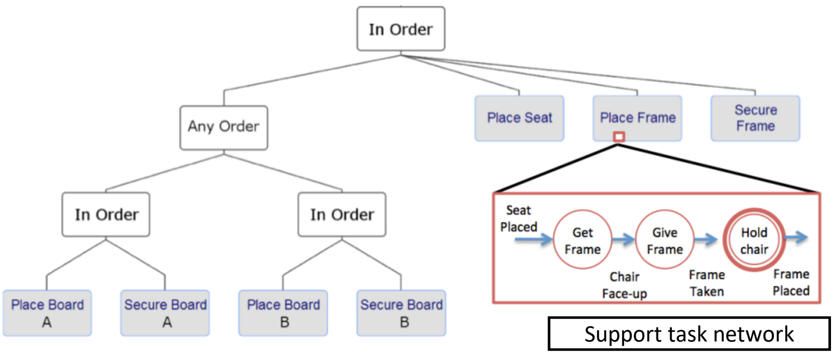
Key Insight:

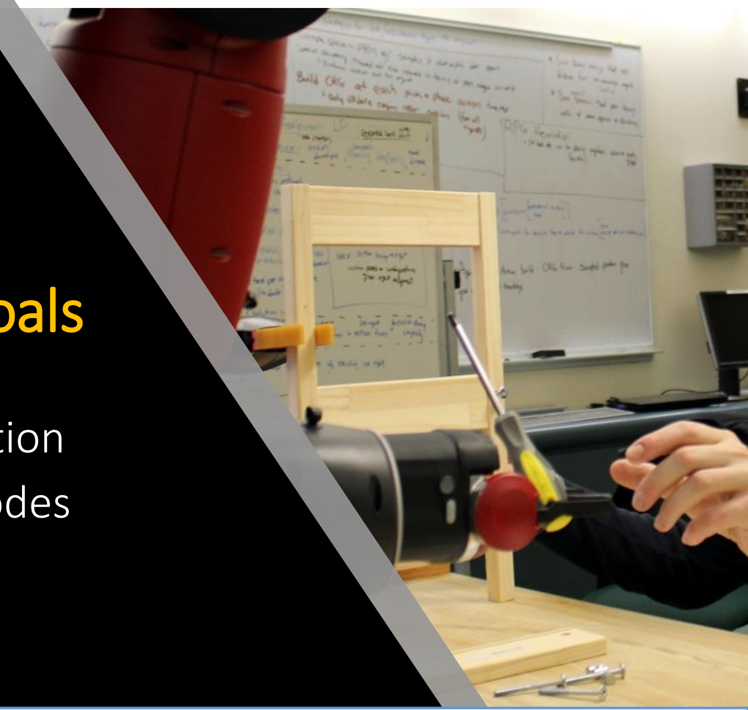
- Apply outlier detection methods across internal activity classifiers
- Use outliers or lack thereof to explain issues across **time** and **objects**

Asking a “carry” classifier about a “walk” trajectory:

“In the **middle and end** of the trajectory, the **left hand and right hand** features were very poorly matched to my template.”

Supportive Behaviors by Demonstration





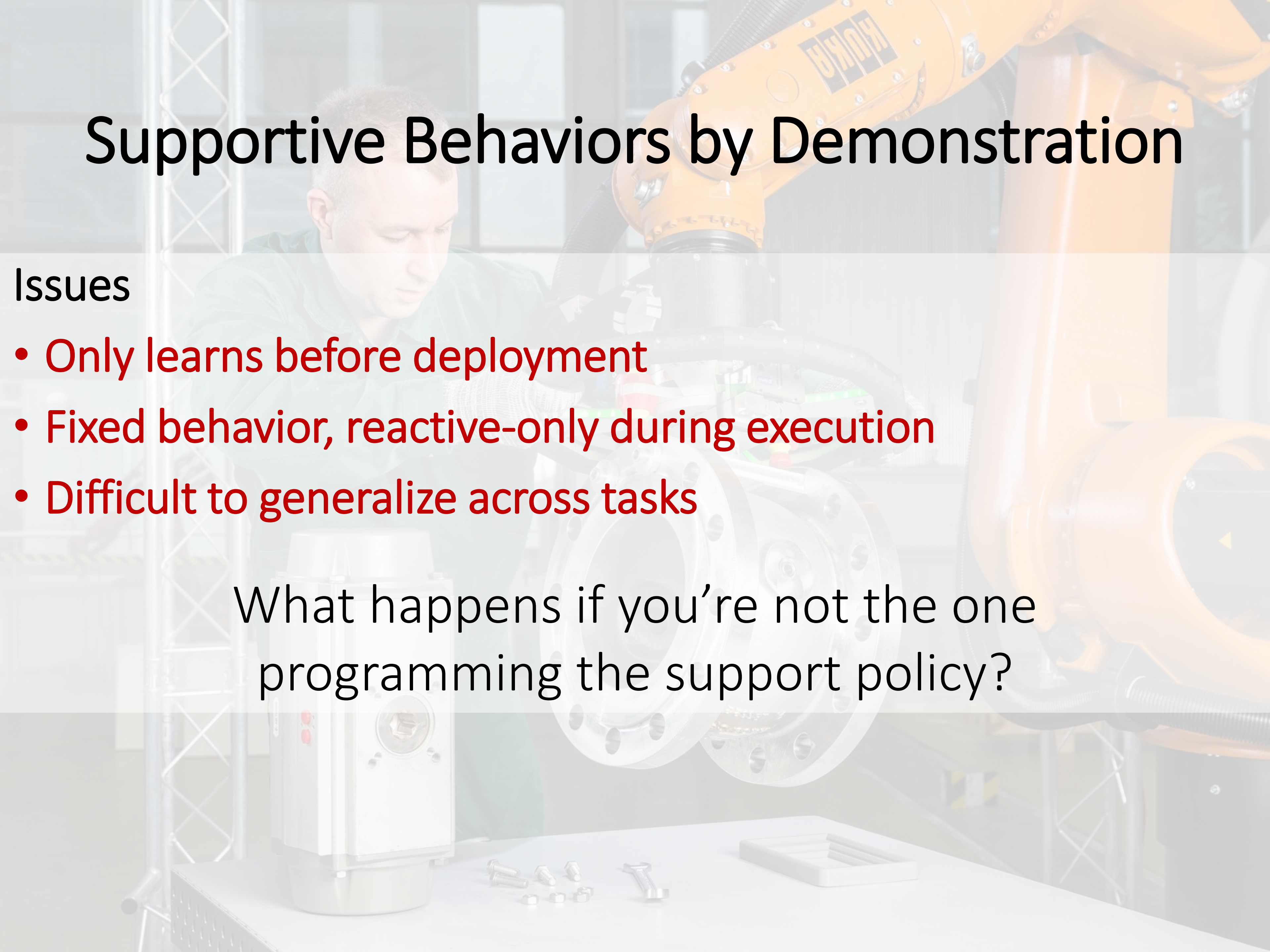
Associating **supportive behaviors** with **subgoals**

Explicitly learned from demonstration during task execution

Support policy can be propagated to higher-level task nodes

Hayes & Scassellati, "Online Development of Assistive Robot Behaviors for Collaborative Manipulation and Human-Robot Teamwork", Machine Learning for Interactive Systems, AAAI 2014

Context-sensitive Supportive Behavior Policies



Supportive Behaviors by Demonstration

Issues

- Only learns before deployment
- Fixed behavior, reactive-only during execution
- Difficult to generalize across tasks

What happens if you're not the one programming the support policy?

Learning from Demonstration Breaks Down in Team Scenarios!

Traditional LfD is optimal if the reference demonstrations are “Expert” demonstrations.

…but execution happens in isolation!

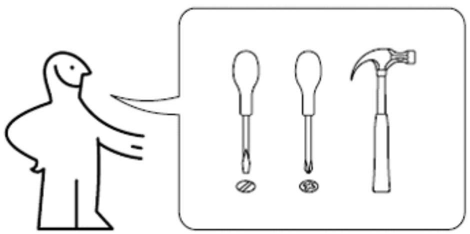
Expert demonstrations are not always the most effective teaching strategy.

Sometimes it’s better to learn the landscape of the problem than to see optimal demonstrations

Properly crafted ‘imperfect’ demonstrations can better communicate information about the objective.

Leading to one all-important question…

Can we do better than learning from examples?



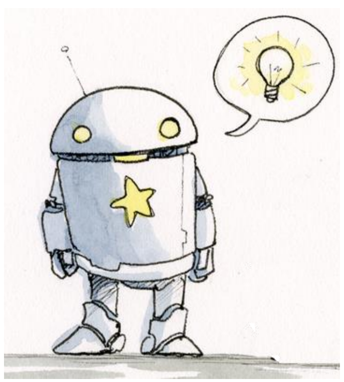
Demonstration-based Methods

Human figures out ***how*** and ***when*** the robot can be helpful

\+ Quickly enables useful, helpful actions.

− Does not scale with task count!

− Requires human expert



Goal-driven Methods

Robot figures out ***how*** and ***when*** it can be helpful

Allows for novel behaviors to be discovered

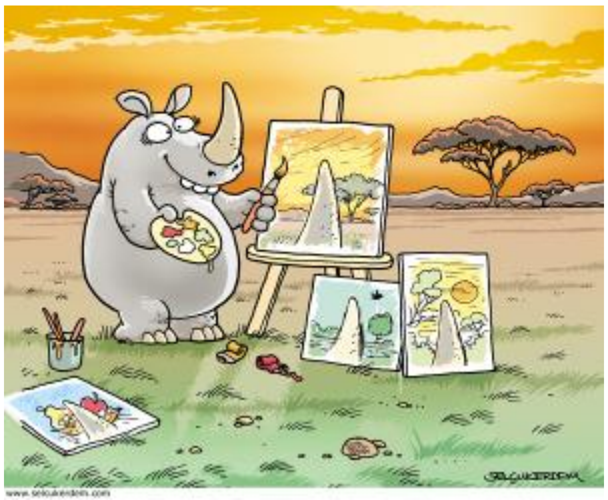
Enables deeper task comprehension and action understanding

Effective Robot Teammate Behaviors for Supporting Sequential Manipulation Tasks

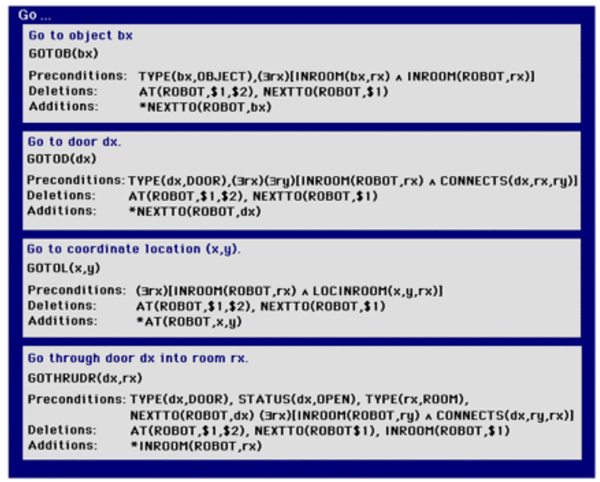
[IROS 15]

Bradley Hayes and Brian Scassellati

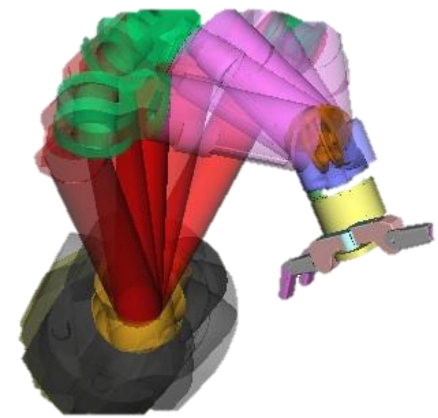
Autonomously Generating Supportive Behaviors:
A Task and Motion Planning Approach



Perspective Taking



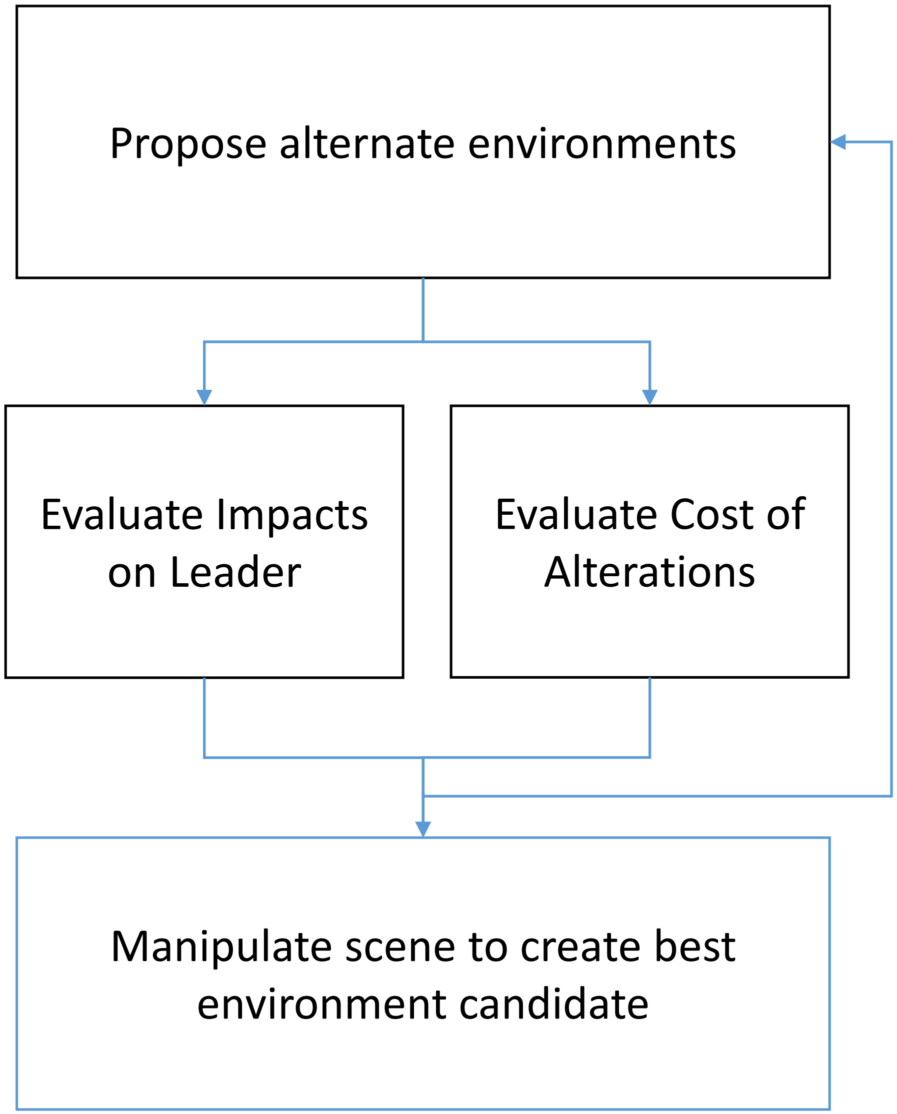
Symbolic planning



Motion planning

Autonomously Generated Supportive Behaviors

Supportive Behavior Pipeline: Intuition



1. Propose alternative environments
   - Change **one** thing about the environment
2. Evaluate if they facilitate the leader's task/motion planning
   - Simulate policy execution(s) from leader's perspective
3. Compute cost of creating target environment
   - Simulate support agent's plan execution
4. Choose environment that maximizes [benefit – cost]
   - Execute supportive behavior plan

Plan Evaluation

Choose the support policy (**ξ** ∈ **Ξ**) that minimizes the expected execution cost of the leader's policy (**π** ∈ **Π**) to solve the TAMP problem **T** from the current state (**s**$_{\mathbf{c}}$)

- Cost estimate must account for
  - Resource conflicts (shared utilization/demand)
  - Spatial constraints (support agent's avoidance of lead)

$$\min_{\xi \in \Xi} \sum_{\pi \in \Pi_T} w_\pi * \text{cost}\left(T, \pi, \xi, s_c, \gamma\right)$$

Plan Evaluation

Choose the support policy (**ξ** ∈ **Ξ**) that minimizes the expected execution cost of the leader's policy (**π** ∈ **Π**) to solve the TAMP problem **T** from the current state (**s$_c$**)

- Cost estimate
  - Resource ... ation/demand)
  - Spatial co ... nt's avoidance of lead)

Weighting function makes a big difference!

$$\min_{\xi \in \Xi} \sum_{\pi \in \Pi_T} w_\pi * \text{cost}\left(T, \pi, \xi, s_c, \gamma\right)$$

Weighting functions: Uniform, Greedy

$$w_\pi = 1$$

Consider all known solutions equivalently likely and important

$$w_\pi = \begin{cases} 1 & ; \quad \text{duration}(T, \pi, \emptyset, s_0, f(x) = 1) = \text{Min duration} \\ 0 & ; \quad \text{otherwise} \end{cases}$$

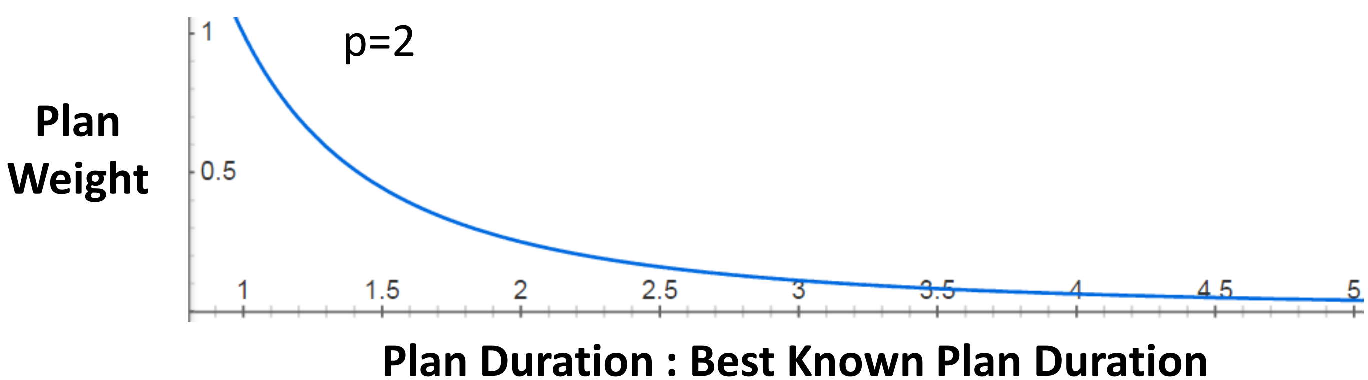
Only the best-known solution is worth planning against

Weighting functions: Uniform

Weighting functions: Optimality-Proportional

$$w_\pi = \left(\frac{\min\limits_{\pi \in \Pi_T} \text{duration}(T, \pi, \emptyset, s_0, f(x) = 1)}{\text{duration}(T, \pi, \emptyset, s_0, f(x) = 1)} \right)^p$$

Weight plans proportional to their cost vs. the best-known solution



Weighting functions: Error Mitigation

$$w_\pi = \begin{cases} f(\pi) & ; \ \text{duration}(T, \pi, \emptyset, s_0, f(x) = 1) \leq \epsilon \\ -\alpha w_\pi & ; \ \text{otherwise} \end{cases}$$

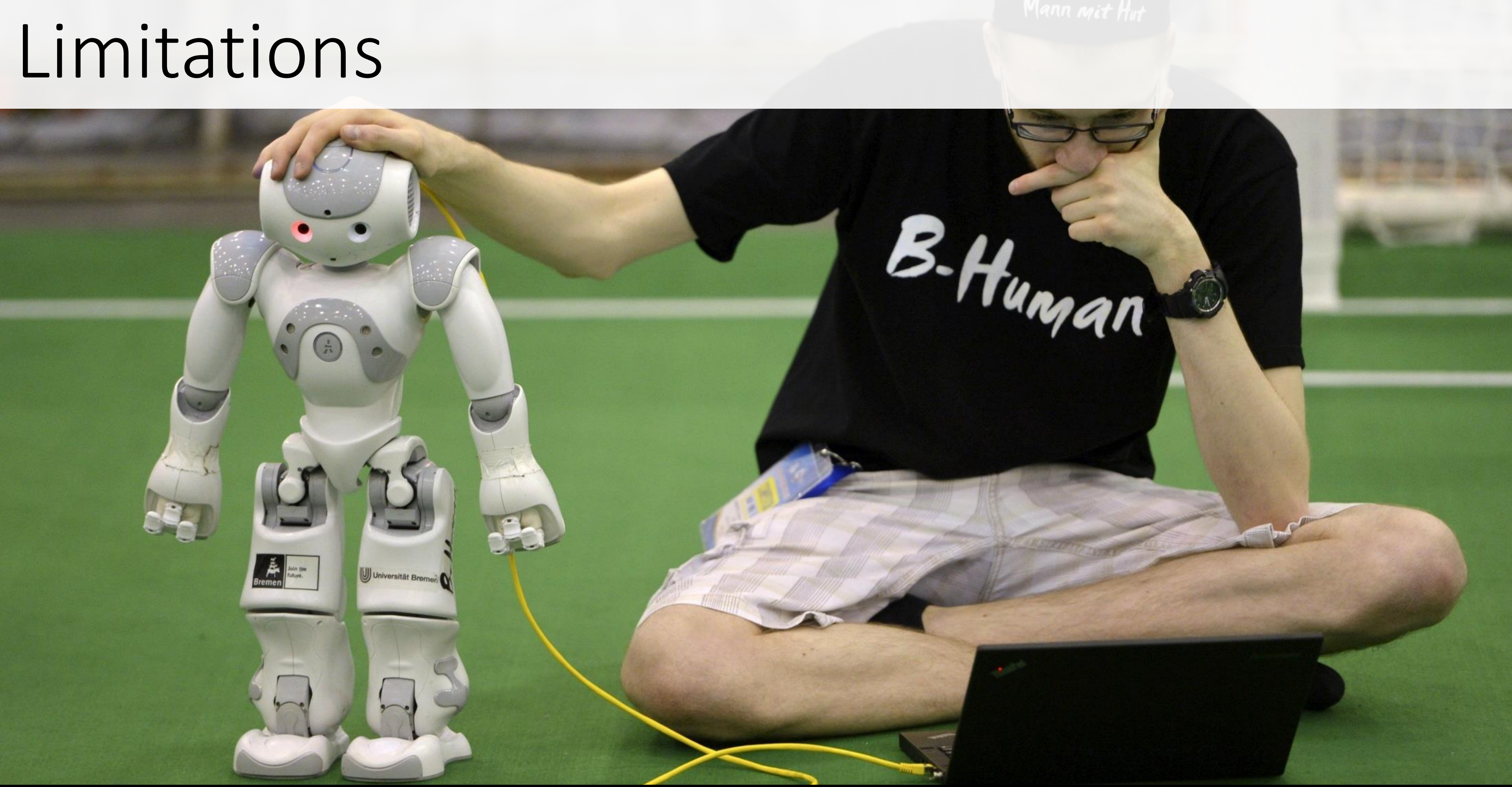
Plans more optimal than some cutoff ε are treated normally, per *f*.

Suboptimal plans are negatively weighted, encouraging active mitigation behavior from the supportive robot.

$\alpha < \frac{1}{\max_\pi w_\pi}$ is a normalization term to avoid harm due to plan overlap

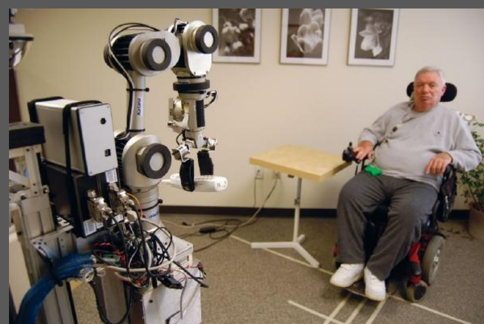
Weighting functions: Error Mitigation

Limitations



- Short forward lookahead (<10 seconds)
- Sampling problem is incredibly difficult
  - Pushes some of the same problems that LfD has into the sampling mechanism
- A priori knowledge of human policy space is necessary
  - This is coordination, not planning!

The Promise of Collaborative Robots



The Reality of Mismatched Expectations

Improving Robot Controller Transparency Through Autonomous Policy Explanation

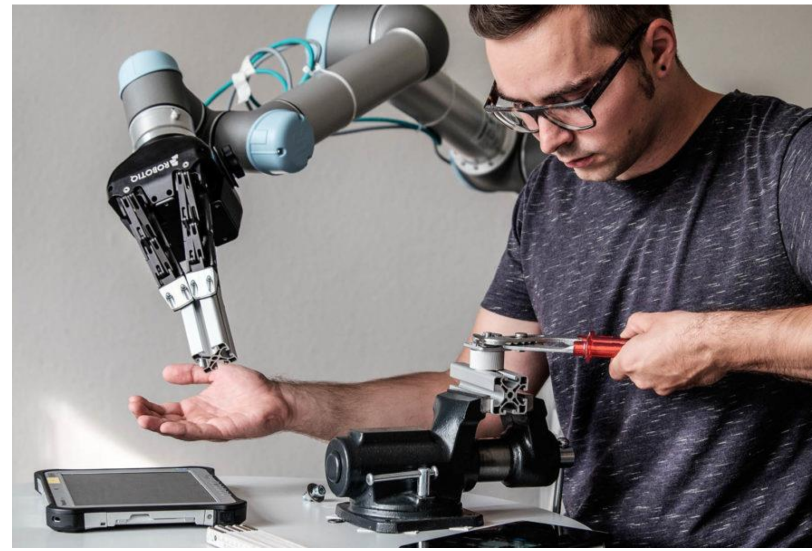
[HRI 17]

Bradley Hayes and Julie Shah

Shared Expectations are Critical for Teamwork

In close human-robot collaboration…

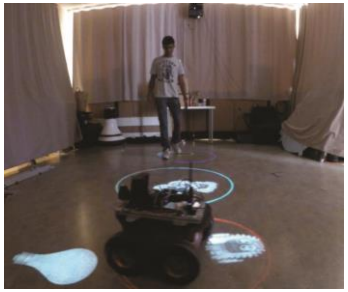
- Human must be able to plan around expected robot behaviors
- Understanding failure modes and policies are central to ensuring safe interaction and **managing risk**



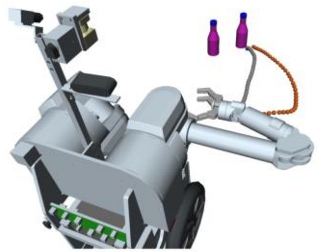
Fluent teaming **requires** communication…

- When there’s no prior knowledge
- When expectations are violated
- When there is joint action

Establishing Shared Expectations



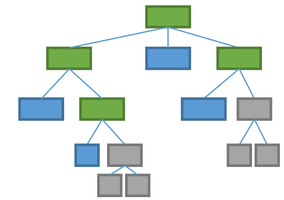
Role-based Feedback
[St. Clair et al. 2016]



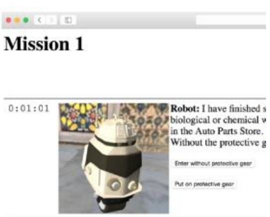
Legible Motion
[Dragan et al. 2013]

	Break	Idle	Negotiate	Sell	InnerTalk	Watch	Guard	Equip
Anny	0	0	0	1	1	1	0	0
Benny	0	0	0	1	1	1	0	0
Canny	1	0	0	0	0	0	0	1
Danny	0	0	1	1	1	0	0	0
Erny	0	1	0	0	0	0	1	0
Frenny	1	0	0	0	0	0	0	1

Coordination Graphs
[Kalech 2010]

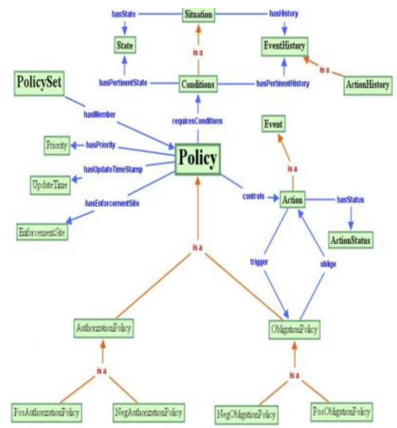


Hierarchical Task Models
[Hayes et al. 2016]

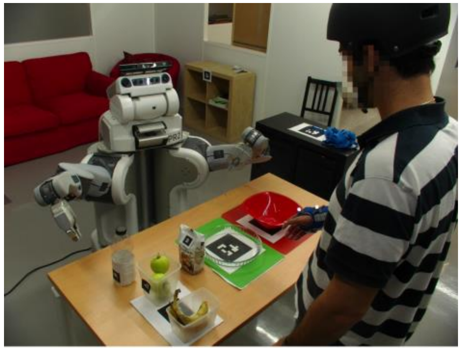


State Disambiguation
[Wang et al. 2016]

Cross-training
[Nikolaidis et al. 2013]



Policy Dictation
[Johnson et al. 2006]



Collaborative Planning
[Milliez et al. 2016]

Short Term

Long Term

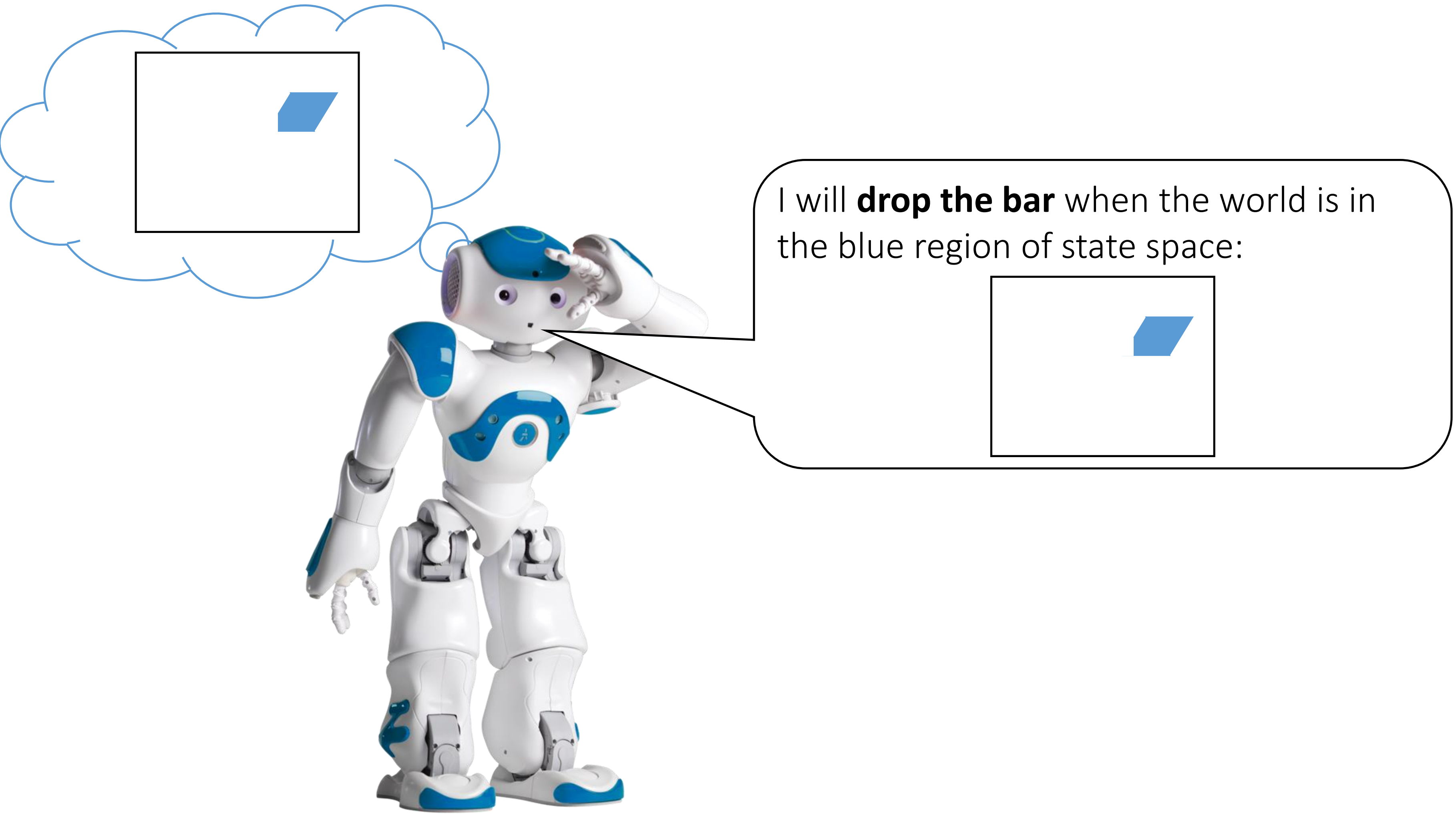
Semantics for Policy Transfer

Under what conditions will you drop the bar?

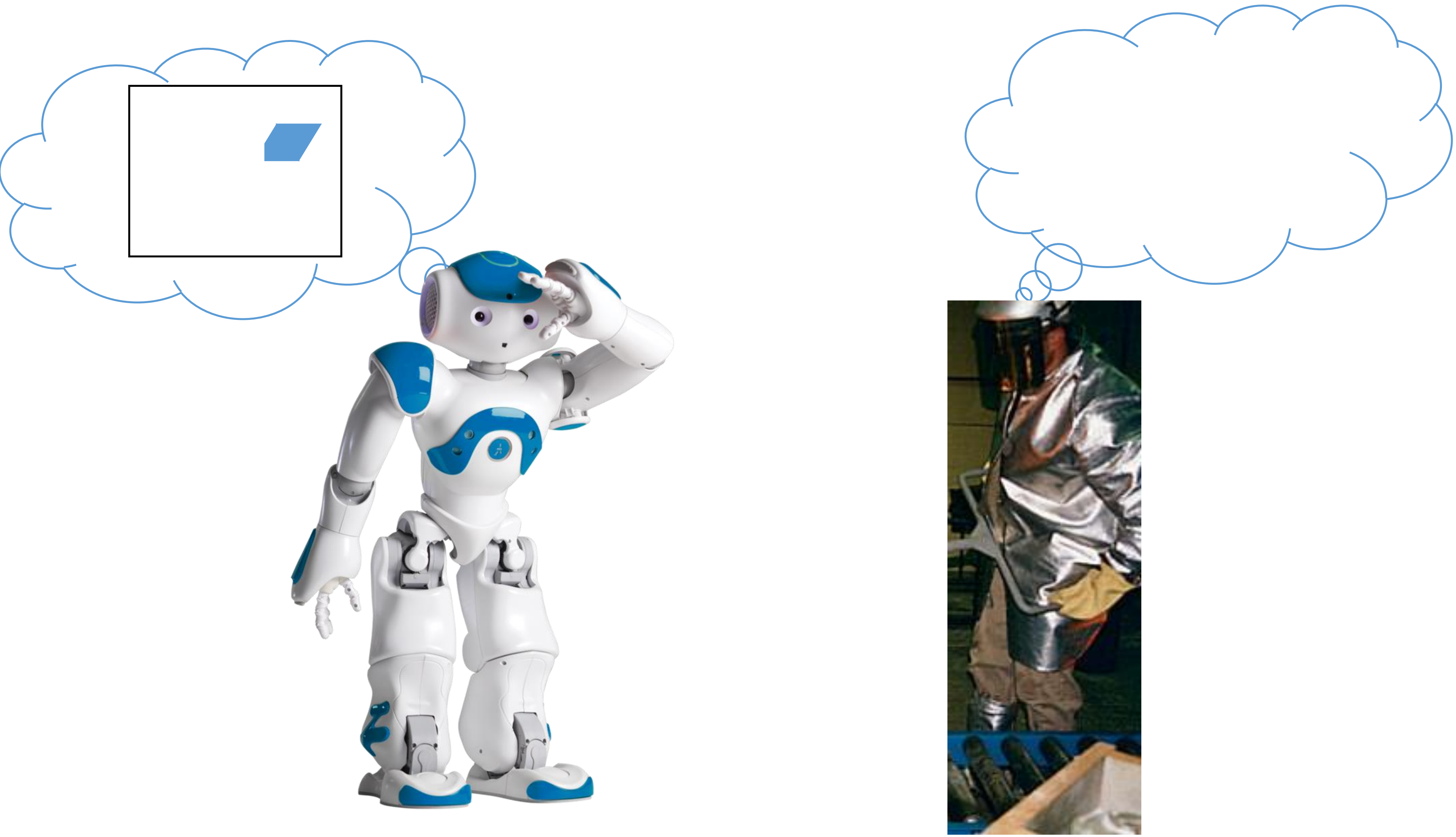
Semantics for Policy Transfer

Under what conditions will you drop the bar?

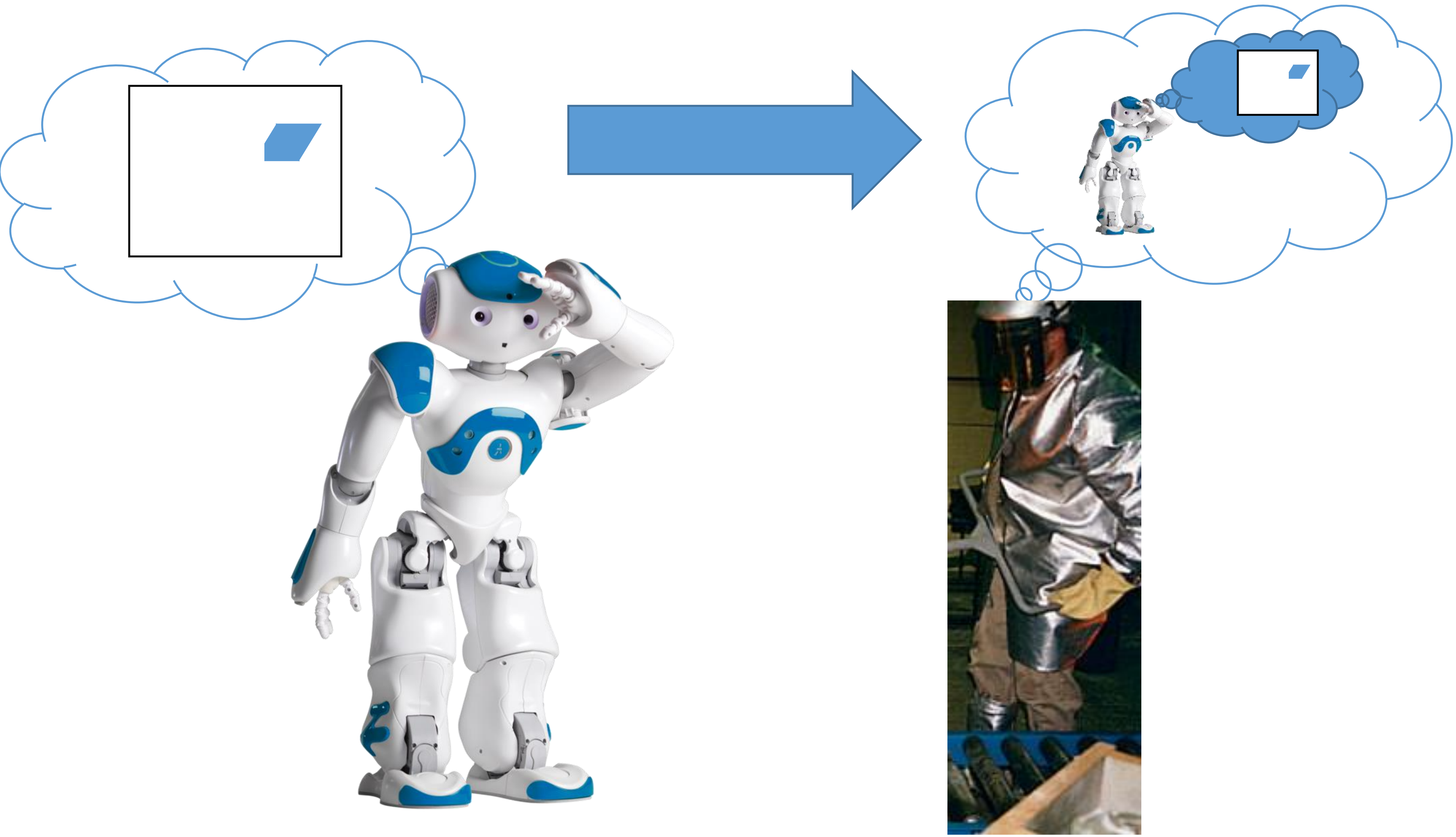
Semantics for Policy Transfer

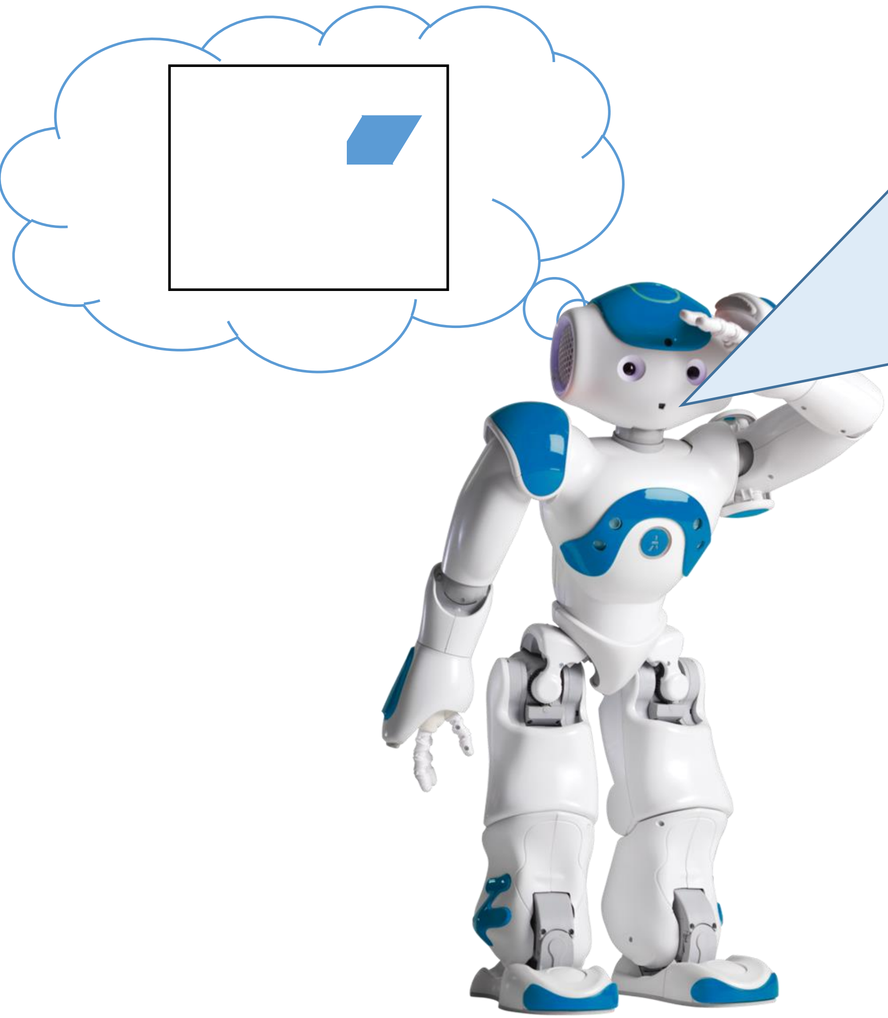


Semantics for Policy Transfer



Semantics for Policy Transfer





I will **drop the bar** when the world is in the blue region of state space:

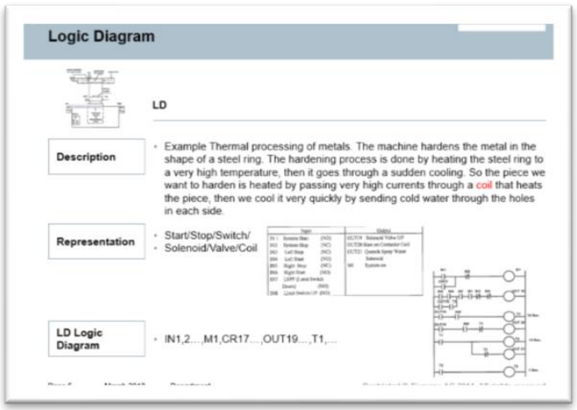
$$\begin{pmatrix}12.4827\\5.12893\\1.12419\\0\\0\\1\\3.62242\\-40.241\\\dots\end{pmatrix}, \begin{pmatrix}15\\7.125\\1.12419\\0\\0\\1\\-8.1219\\-40\\\dots\end{pmatrix}, \begin{pmatrix}12.4827\\8.51422\\1.12419\\0\\1\\0\\3.62242\\-40.241\\\dots\end{pmatrix} \dots$$

State space is too obscure to directly articulate

I will **drop the bar** when the world is in the blue region of state space:

$$\begin{pmatrix} 12.4827 \\ 5.12893 \\ 1.12419 \\ 0 \\ 0 \\ 1 \\ 3.62242 \\ -40.241 \\ \dots \end{pmatrix}, \begin{pmatrix} 15 \\ 7.125 \\ 1.12419 \\ 0 \\ 0 \\ 1 \\ -8.1219 \\ -40 \\ \dots \end{pmatrix}, \begin{pmatrix} 12.4827 \\ 8.51422 \\ 1.12419 \\ 0 \\ 1 \\ 0 \\ 3.62242 \\ -40.241 \\ \dots \end{pmatrix} \dots$$

State of the Art



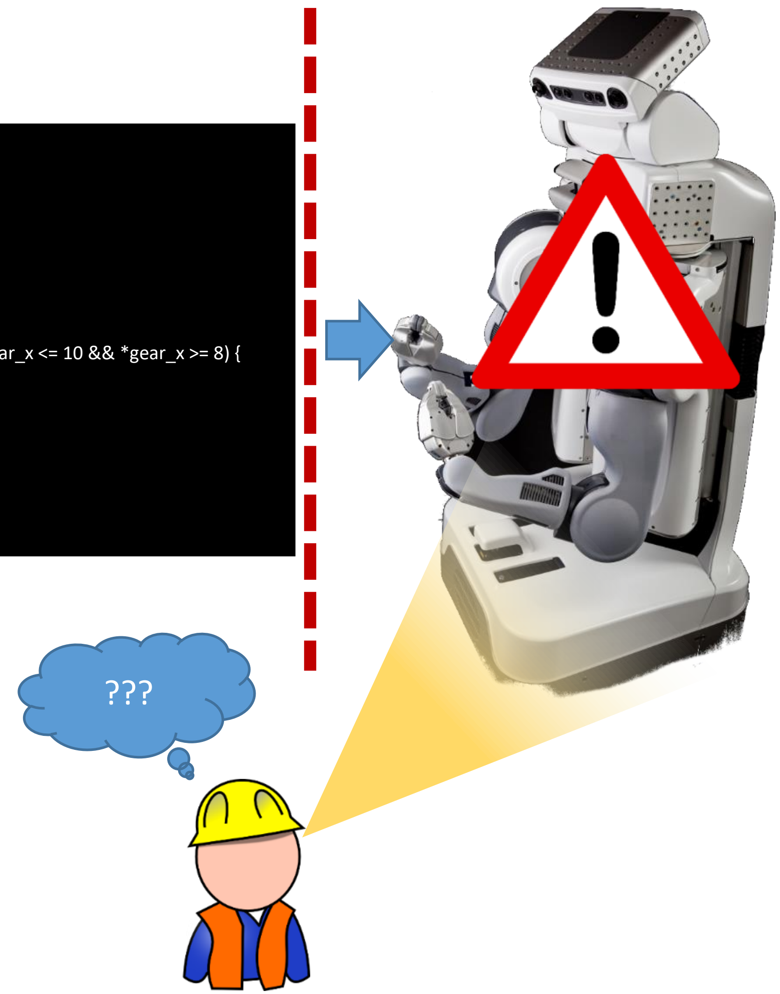
Logic Diagram

LD

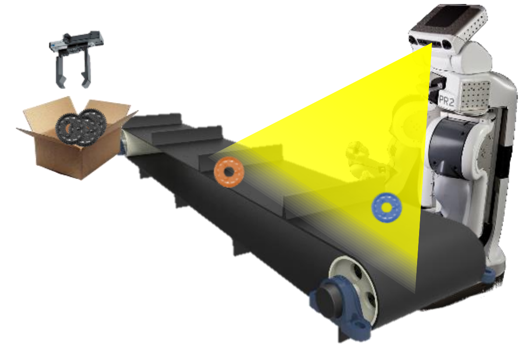
Description	• Example Thermal processing of metals. The machine hardens the metal in the shape of a steel ring. The hardening process is done by heating the steel ring to a very high temperature, then it goes through a sudden cooling. So the piece we want to harden is heated by passing very high currents through a coil that heats the piece, then we cool it very quickly by sending cold water through the holes in each side.
Representation	• Start/Stop/Switch/ • Solenoid/Valve/Coil
LD Logic Diagram	• IN1,2…,M1,CR17…,OUT19…,T1,…

```
int *detect_gear = &INPUT1;
int *gear_x = &INPUT2;

if (*detect_gear == 1 && *gear_x <= 10 && *gear_x >= 8) {
    pick_gear(gear_x);
}
```



Natural Interaction



Reasonable question:

“Why didn’t you inspect the gear?”

Interpretable answer:

“My camera didn’t see a gear. I inspect the gear when it is less than 0.3m from the conveyor belt center and it has been placed by the gantry.”

Fault Diagnosis

Policy Explanation

Root Cause Analysis

Making Control Systems More Interpretable

Approach:

1. Attach a smart debugger to monitor controller execution
2. Build a graphical model from observations

Model Building

3. Use specialized algorithms to map queries to state regions
4. Collect relevant state region attributes
5. Minimally summarize relevant state regions with attributes

Query Analysis

6. Communicate query response

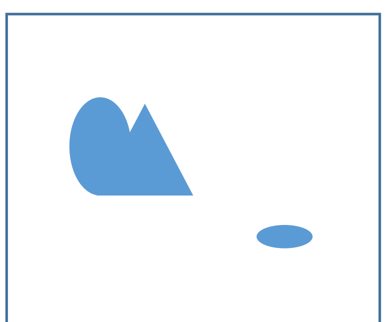
Response Generation

Concept Representations

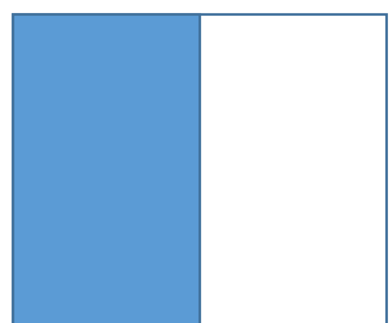
Concept library: generic state classifiers mapped to semantic templates that identify whether a state fulfills a given criteria

Set of Boolean classifiers: State → {True, False}

- Spatial concepts (e.g., “A is on top of B”)
- Domain-specific concepts (e.g., “Widget paint is drying”)
- Agent-specific concepts (e.g., “Camera is powered”)



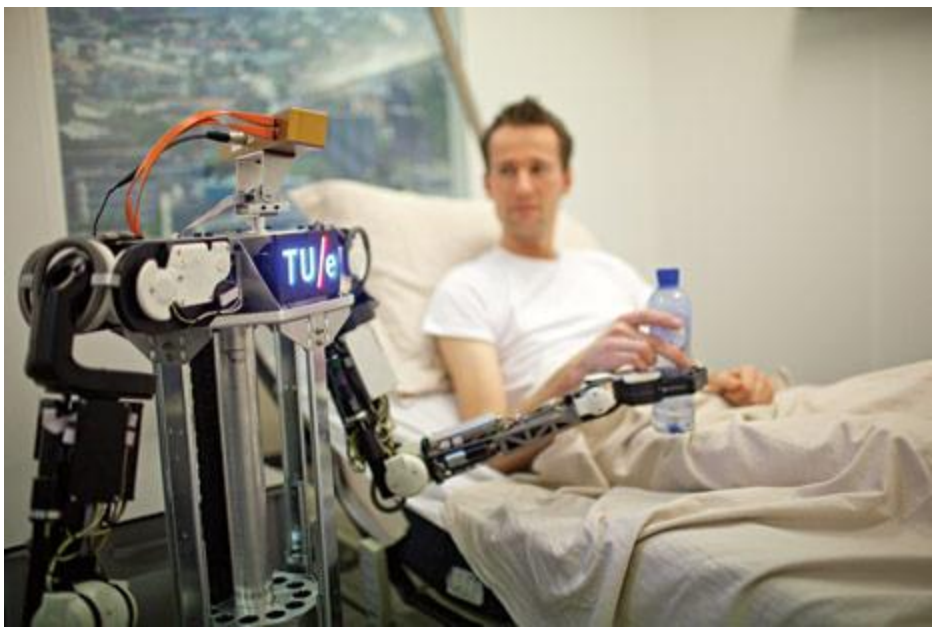
on_top(A,B)



camera_powered

Relevant Question Templates

When will you do {action}?



Algorithm 2: Identify Dominant-action State Region

Input: Behavioral Model $G = \{V, E\}$, Target Action a_t
Output: Set of target states S_{π^a}, Set of non-target states $S_{\pi^* \backslash a}$

$S_{\pi^a} \leftarrow \{\}$;
$S_{\pi^* \backslash a} \leftarrow \{\}$;
foreach $s \in V$ **do**
    $a \leftarrow$ most frequent action executed from s;
    if $a == a_t$ **then** $S_{\pi^a} \leftarrow S_{\pi^a} \cup s$;
    else $S_{\pi^* \backslash a} \leftarrow S_{\pi^* \backslash a} \cup s$;
return $S_{\pi^a}, S_{\pi^* \backslash a}$;

Relevant Question Templates

Why didn't you do {action}?

Algorithm 3: Identify Behavioral Divergences

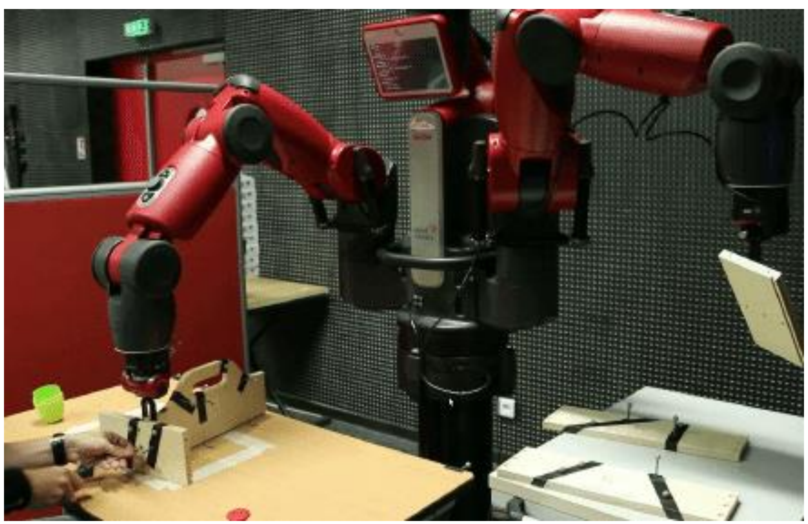
Input: Behavioral Model $G = \{V, E\}$, Target Action a_t, Previous state s_p, Distance threshold D_{const}

Output: Explanation of difference between current state and state region where a_t is performed, explanation of where a_t is performed locally.

$S_{\pi^a} \leftarrow \{\}$;
$S_{\pi^* \setminus a} \leftarrow \{\}$;
foreach $D \in \{1, ..., D_{const}\}$ **do**
  foreach $s \in \{v \in V \mid distance(v, s_p) \leq D\}$ **do**
    $a \leftarrow$ most frequent action executed from s;
    if $a == a_t$ **then** $S_{\pi^a} \leftarrow S_{\pi^a} \cup s$;
    else $S_{\pi^* \setminus a} \leftarrow S_{\pi^* \setminus a} \cup s$;
expected_region $\leftarrow$ describe($G, S_{\pi^a}, S_{\pi^* \setminus a}$);
current_region $\leftarrow$ describe($G, \{s_p\}, S_{\pi^a}$);
return diff(expected_region, current_region), expected_region;

Relevant Question Templates

What will you do when {conditions}?



Algorithm 4: Characterize Situational Behavior

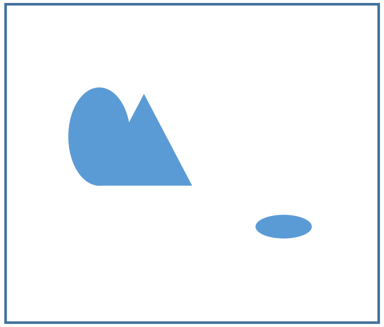
Input: Behavioral Model $G = \{V, E\}$, Concept Library C, State region description d, Max action threshold $cluster\_max$

Output: Explanation of behavior in d, broken down by action and accompanying state region

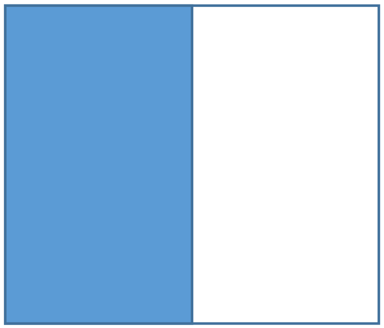
$S \leftarrow dict()$;
$descriptions \leftarrow dict()$;
DNF_description $\leftarrow$ convert_to_DNF_formula(d, C);
foreach $s \in \{v \in V \mid$ *test_dnf(v,DNF_description) is True* $\}$ **do**
  $S[\pi(s)] \leftarrow S[\pi(s)] \cup s$;
  if $|S| > cluster\_max$ **then**
    return too_many_actions_error
foreach $a \in S$ **do**
  descriptions[a] $\leftarrow$ describe($S[a]$);
return descriptions;

Language Mapping: Model to Response

Recall: Concept library provides dictionary of classifiers that cover state regions



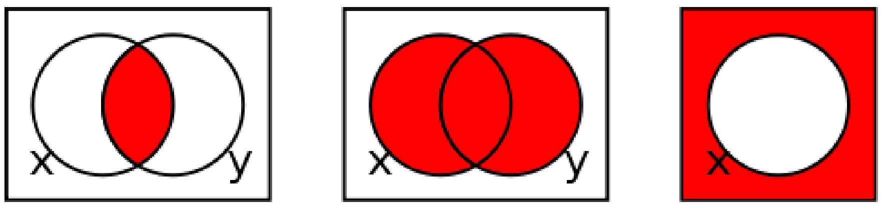
on_top(A,B)



camera_powered

Using Concepts to Describe State Regions

We perform **state-to-language mapping** by applying a Boolean algebra over the space of concepts



$x \wedge y$ $x \vee y$ $\neg x$

This reduces concept selection to a **set cover problem** over state regions

Disjunctive normal form (DNF) formulae enable coverage over arbitrary geometric state space regions via **intersections** and **unions** of concepts

Templates provide a mapping from DNF → natural language

Query Response Process

When do you inspect the gear?

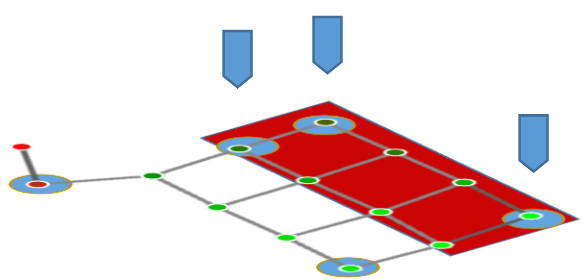
Find states where action {inspect(gear)} is most likely action

Find concept mapping that covers the indicated states

at(conveyor_belt)

Detected_gear

Detected_gear /\ at(conveyor_belt)



Convert to natural language

I'll inspect the gear when I've detected a gear and I'm at the conveyor belt.

Explainable AI Needs Reasoning!

Interpretable and comprehensible systems are lacking in the ability to formulate their line of reasoning, **using human-understandable features of input data**.

Interpretable and comprehensible models **enable** explanations of decisions, but do not yield explanations themselves!

How else can we establish shared expectations and verify that intent was captured?

Have the robot use its model to teach a human!

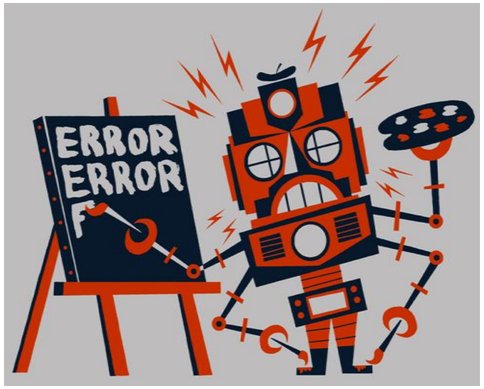
Improving Human-Robot Collaboration through Autonomous Explanation-based Reward Coaching

[HRI 19]

Nominated for Best Technical Paper Award

We spend a lot of time making robots good at things

We're pretty good at this transition



We're less good at this transition

But how do we use this to make others proficient too?

Learning from experience can be expensive

Motivating Questions

How do we turn a capable robot into a competent instructor?

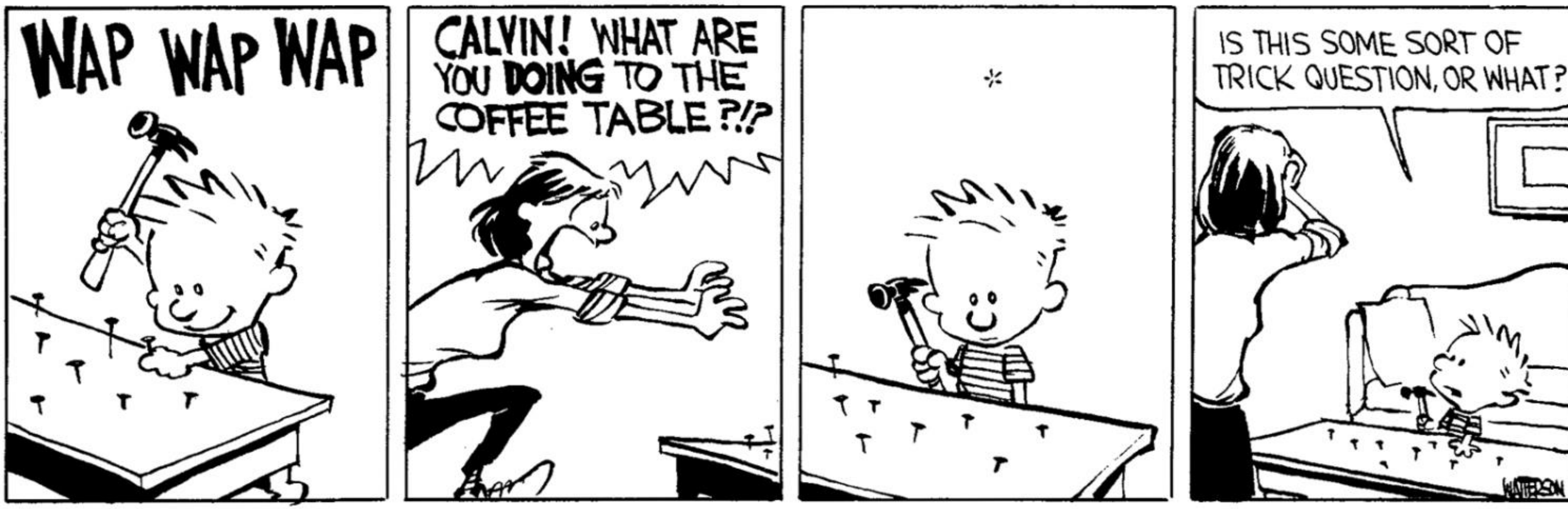
Can a robot use its own understanding of the world to figure out yours?

Given this understanding, can it issue corrective guidance you'll follow?

Can we do all of this within a general framework?

Key Assumption:

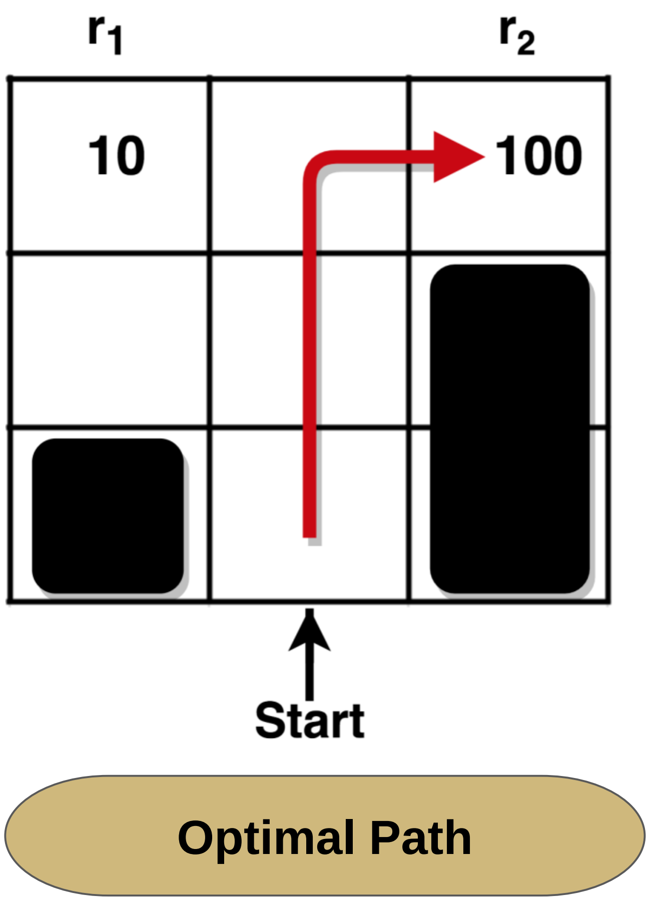
Humans are goal directed, generally rational agents

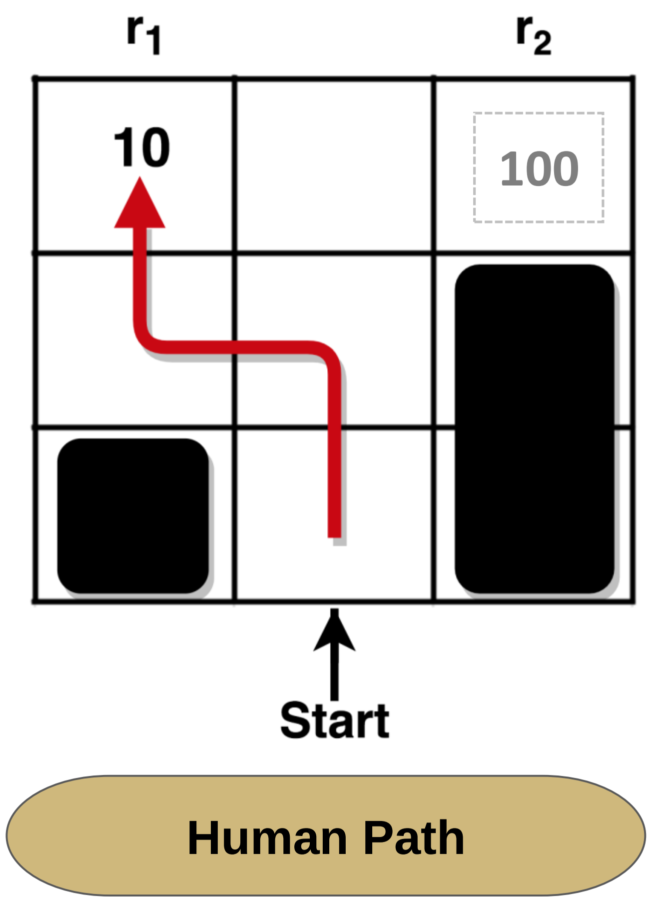


Unexpected policy indicates a difference in (reward function

Humans are agents maximizing their expected reward

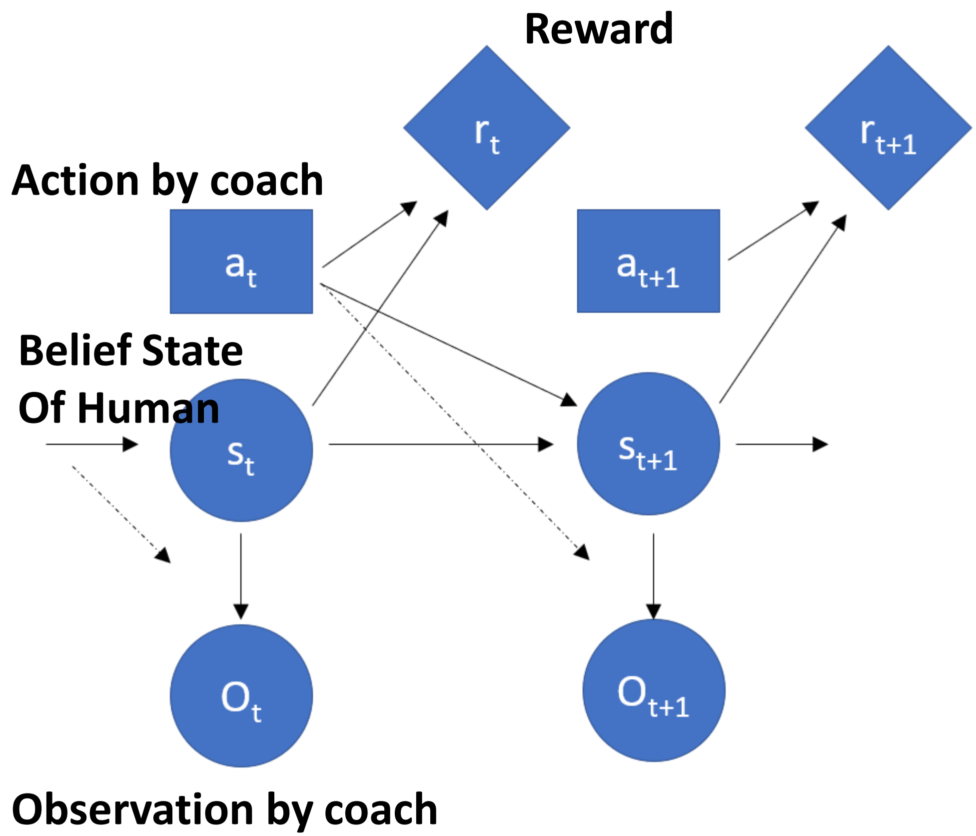
Reward Augmentation and Repair through Explanation





Coaching as **Partially Observable Markov Decision Process**

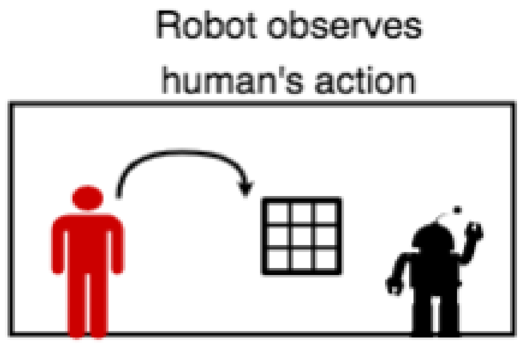
Robot is coaching while collaborating



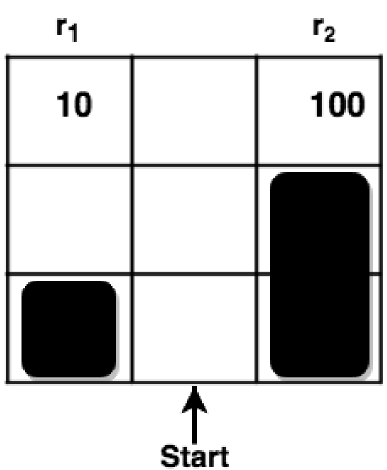
Action can be **task-specific physical action** and **reward repair-specific social action**

RARE: An Intuition

Estimate the collaborator's reward function by figuring out which policy they're following



Assuming policies are optimal w.r.t. the reward function that produced them



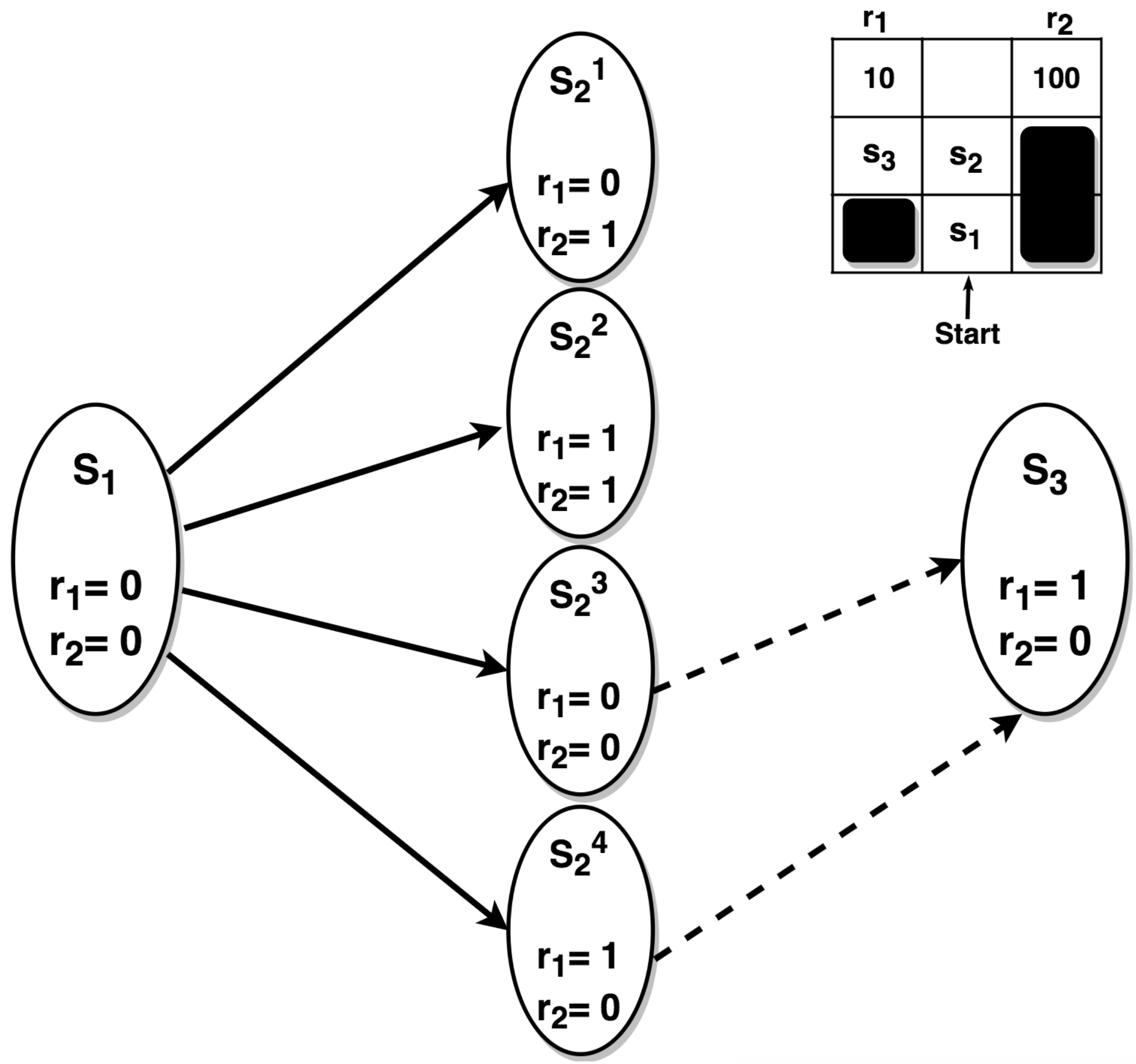
Track belief over reward functions
Using latent Boolean state variables to indicate the collaborator's knowledge about a particular reward.

State Augmentation to Extend Belief and Action Space

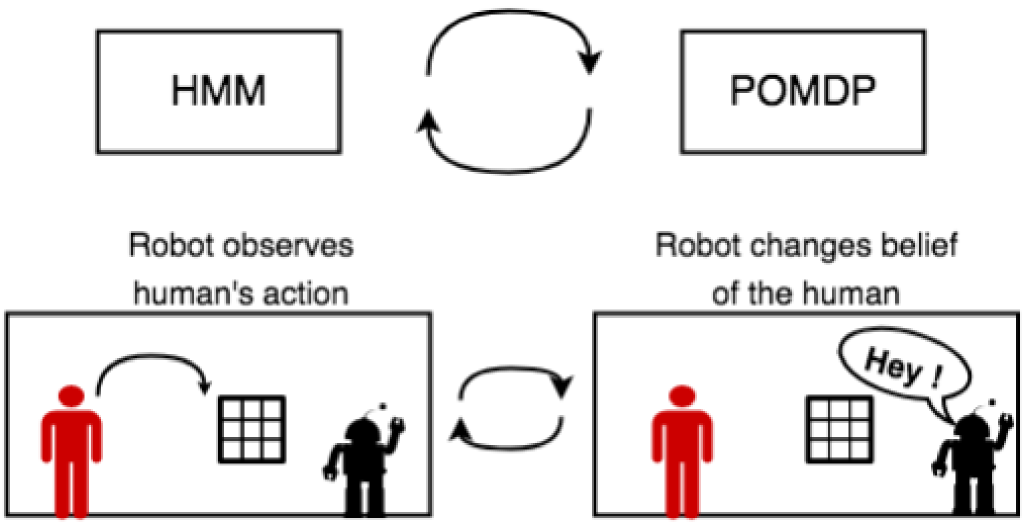
Compound State Vector:

$$S = \begin{bmatrix} W \\ - \\ C \end{bmatrix}, W = \begin{bmatrix} x \\ y \\ \vdots \end{bmatrix} C = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix}$$

World variables
Comprehension variables



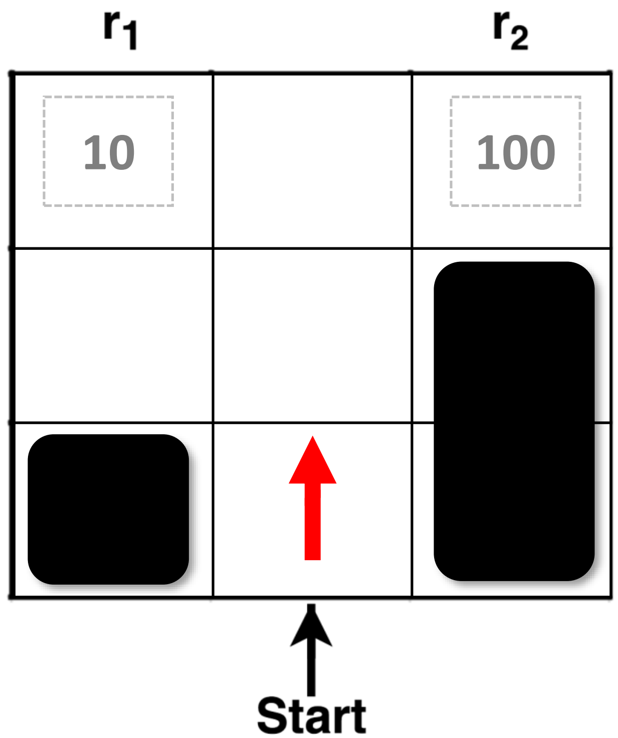
Repairing a Domain Misunderstanding



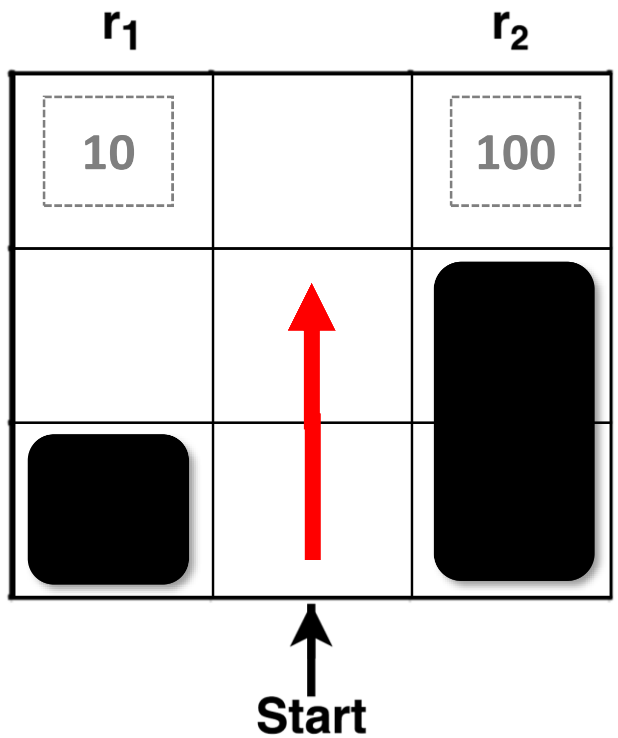
Extend robot's action space

Include communicative actions for revealing reward components.

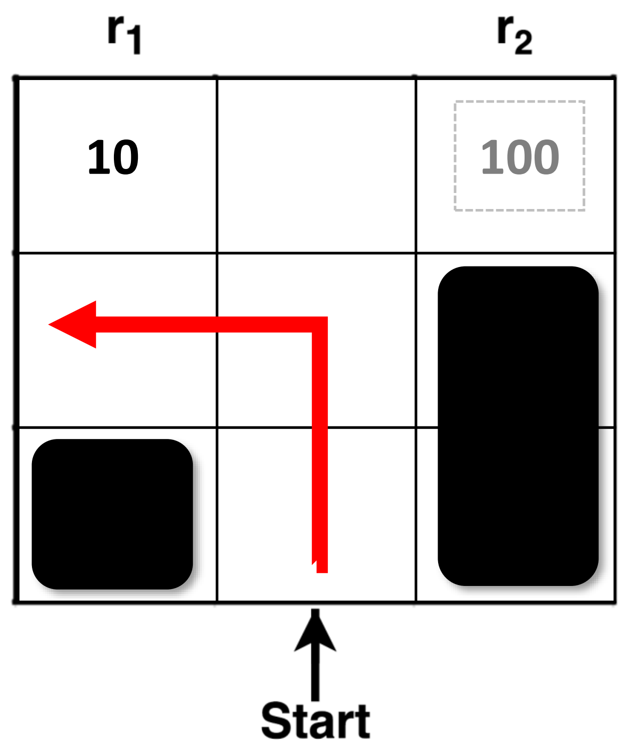
Reward Augmentation through Repair and Explanation



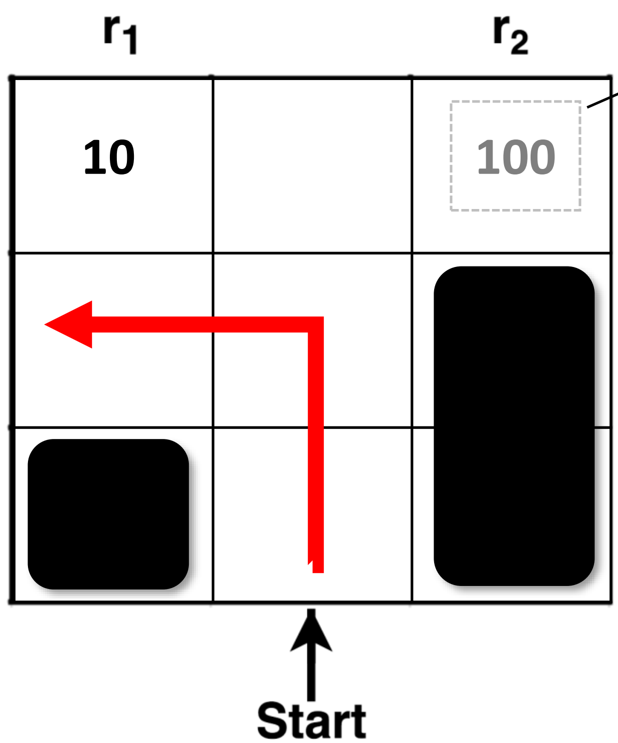
Reward Augmentation through Repair and Explanation



Reward Augmentation through Repair and Explanation

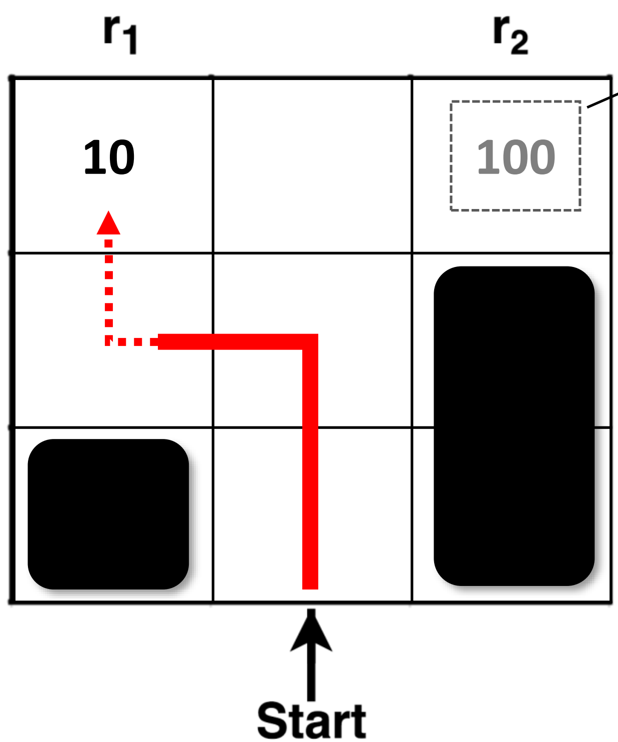


Reward Augmentation through Repair and Explanation



Need to communicate this reward before they finish!

Reward Augmentation through Repair and Explanation

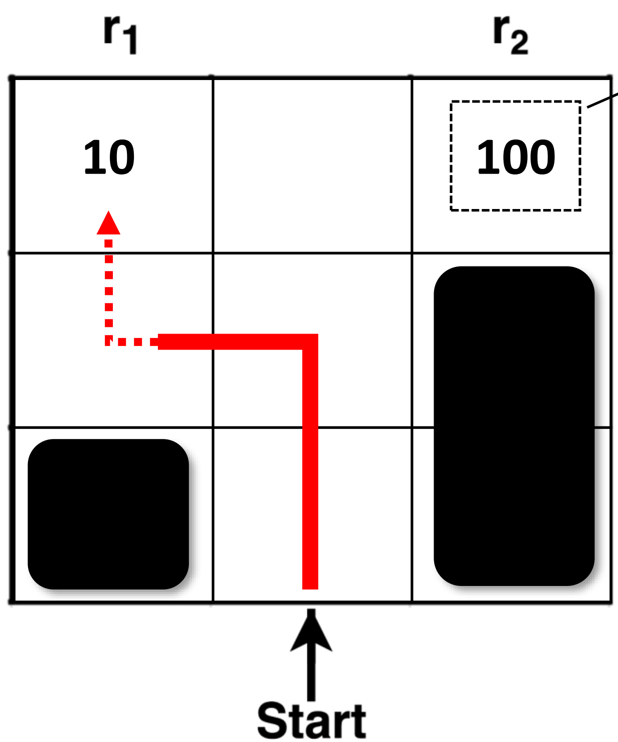


Need to communicate this reward before they finish!

Option 1: "If you do that you won't get the best reward"

Indicate suboptimality of an action to encourage exploration

Reward Augmentation through Repair and Explanation

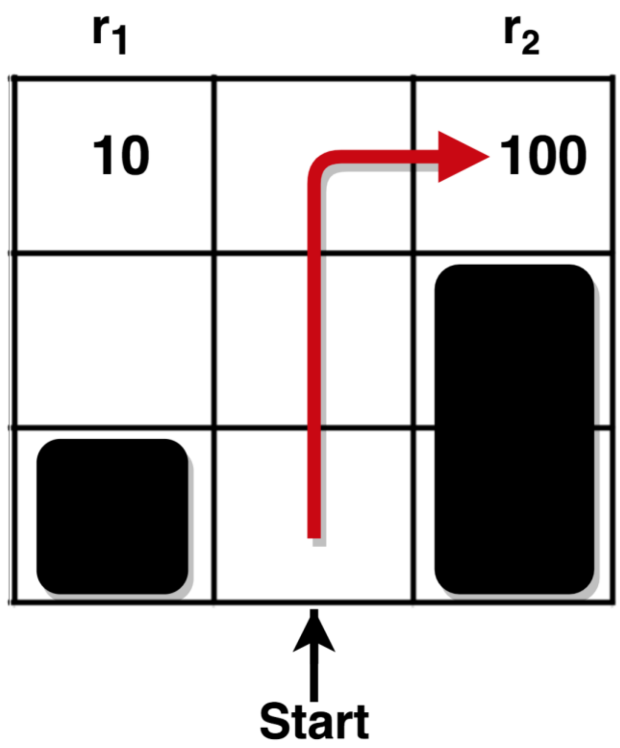


Need to communicate this reward before they finish!

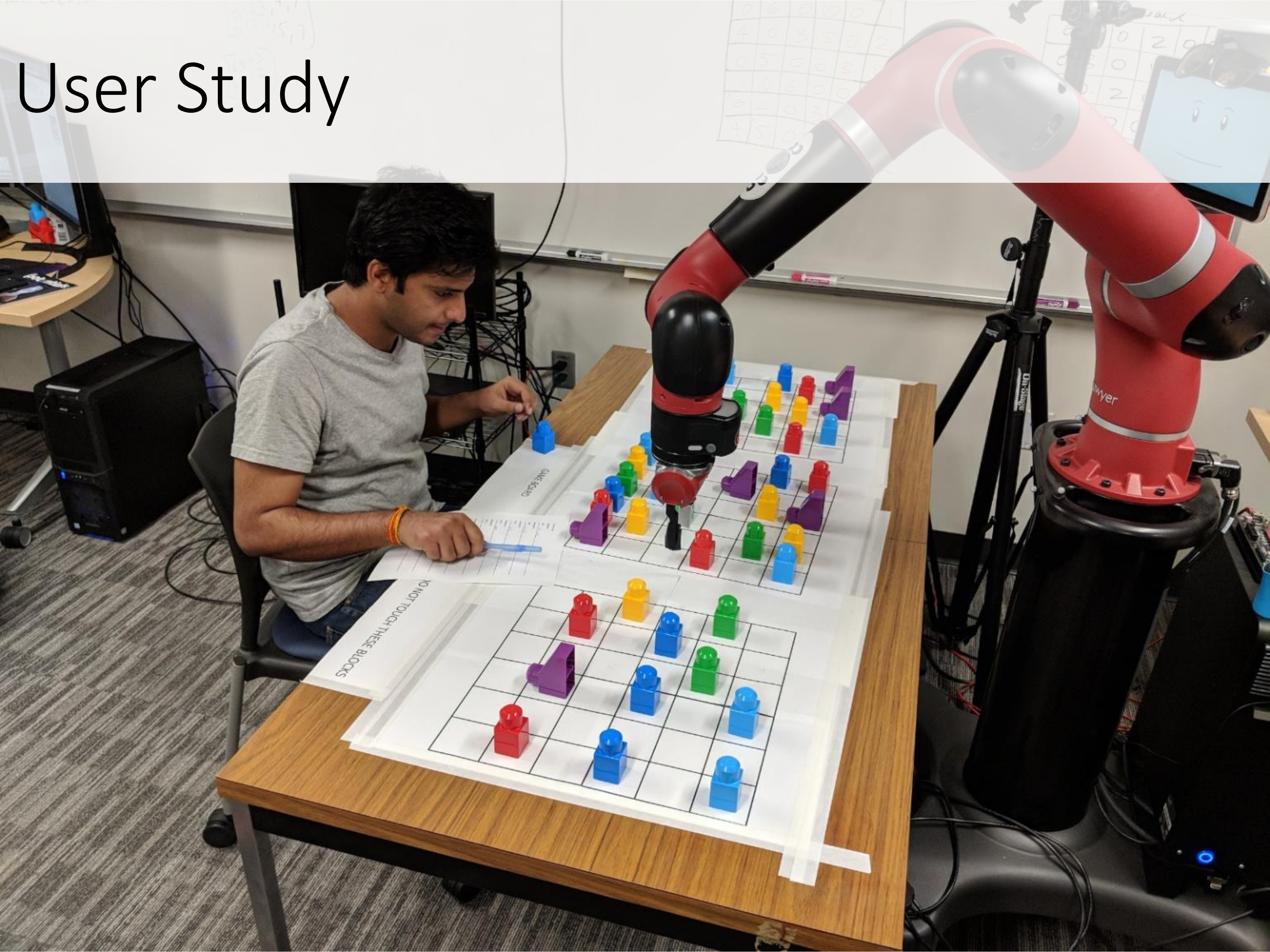
Option 2: “If you do that you won’t get the best reward. There’s a better reward in the top right corner.”

Justify the advice by providing a description of the reward’s location

Reward Augmentation through Repair and Explanation

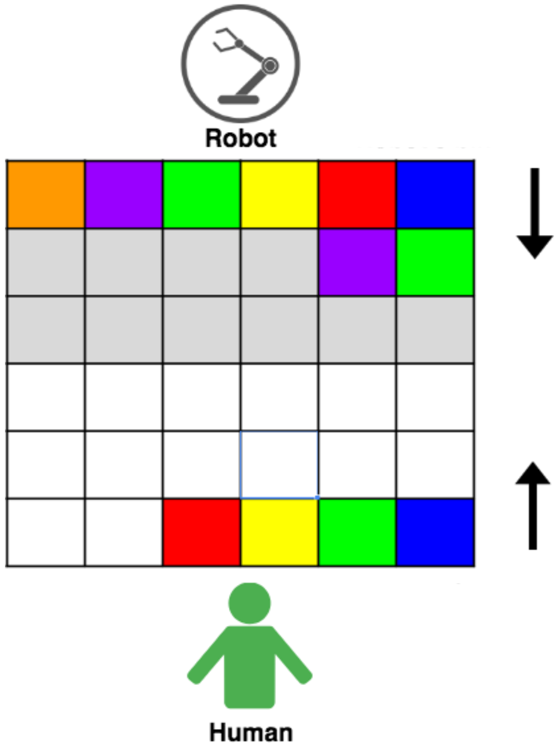


User Study



Realtime Color Sudoku:

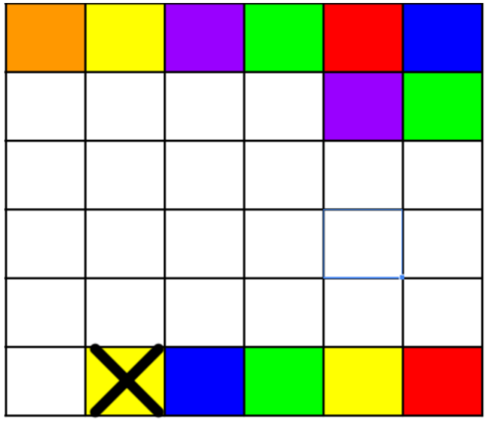
A really hard game for humans



Each player gets 3 rows to fill:
near to far, right to left.

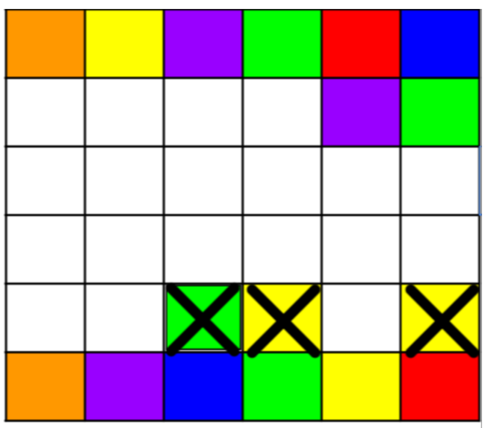
There are no turns:
play whenever you're ready

Realtime Color Sudoku:
The Rules



Row constraint violation

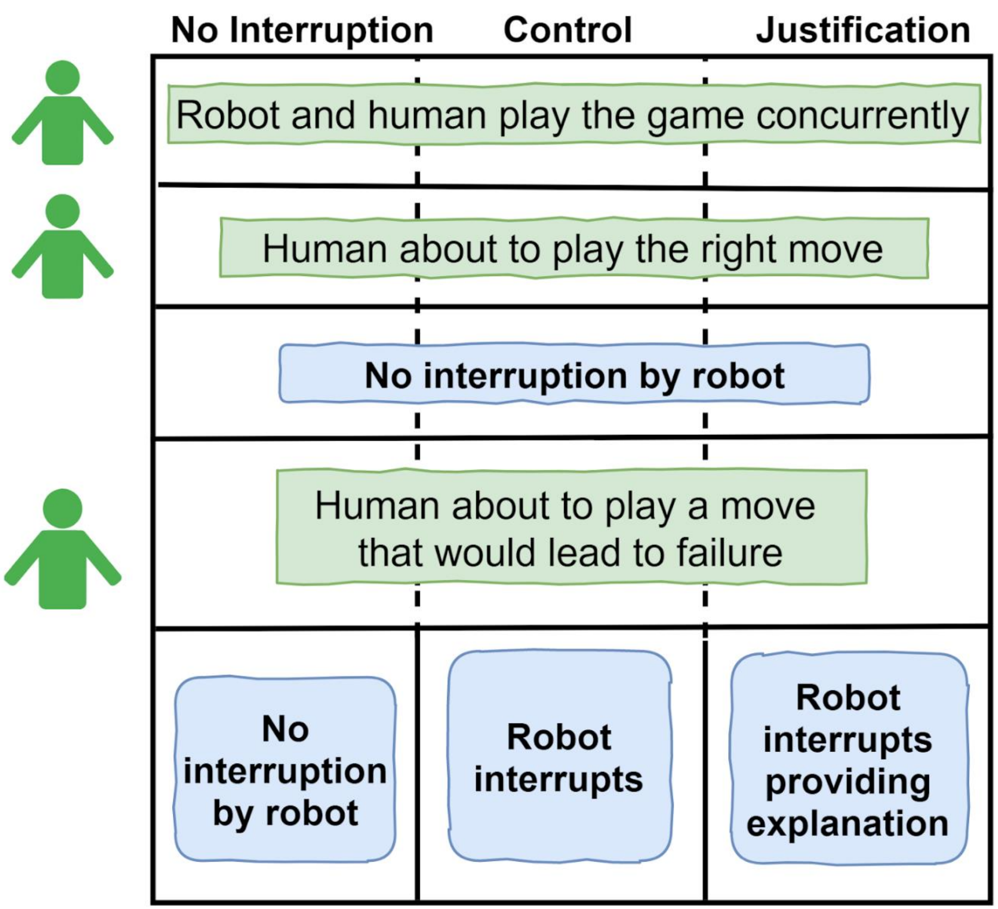
No color may appear twice on the same row



Adjacency constraint violation

No color may border itself

Between-subjects experiment (n=26)



Control:

Players about to make a mistake were told that they cannot make that move or they'll fail the game.

Justification:

Players about to make a mistake were told about the reward inferred they were missing.

No Interruption:

Players completed the game without mistakes.

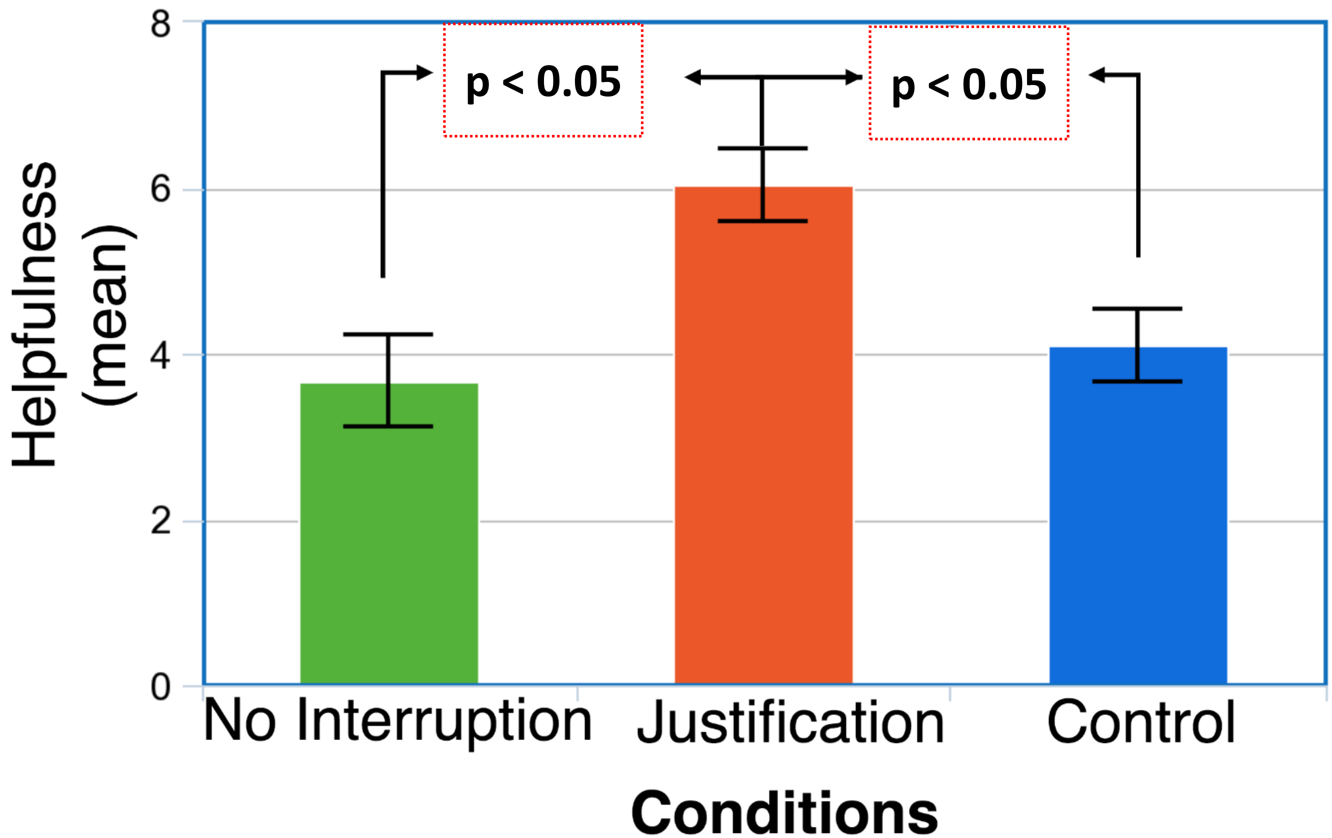
Subjective Hypotheses

H1: **Participants will find the robot more helpful and useful** when it explains why a failure may occur

H2: **Participants will find the robot to be more intelligent** when providing justification for its advice

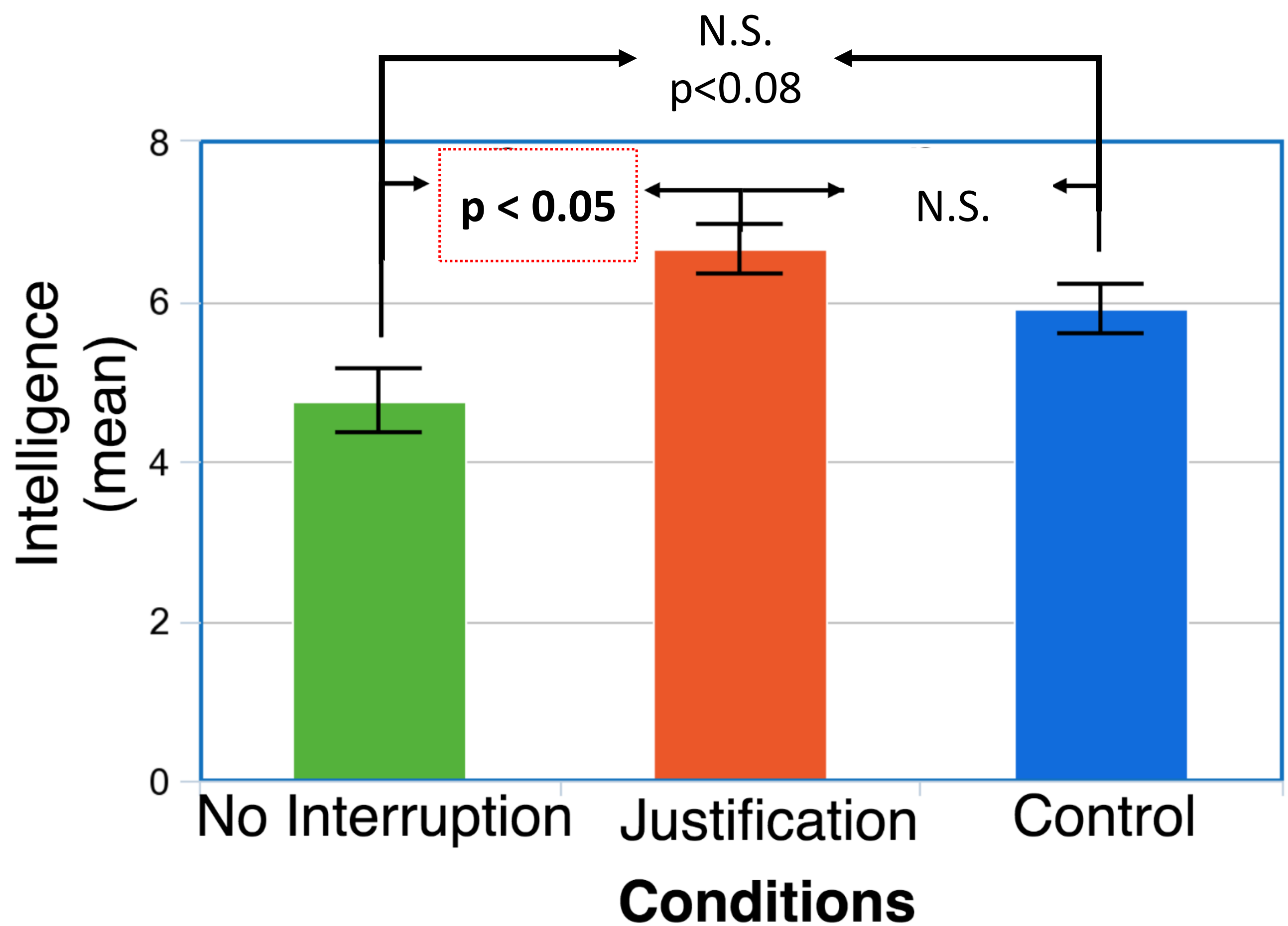
H3: **Participants will find the robot more sociable** when it provides justifications for its failure mitigation

Subjective Results: Helpfulness



H1: **Participants will find the robot more helpful and useful** when it explains why a failure may occur

Subjective Results: Intelligence



H2: **Participants will find the robot to be more intelligent** when it provides justification for its advice

Subjective Hypotheses

H1: **Participants will find the robot more helpful and useful** when it explains why a failure may occur

H2: **Participants will find the robot to be more intelligent** when coaches them

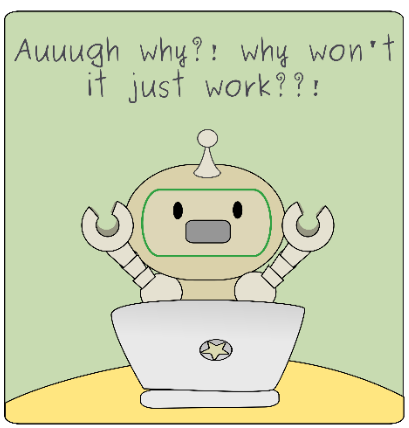
H3: **Participants will find the robot more sociable** when it provides justifications for its failure mitigation

Objective Hypothesis

H1: **Participants will complete the game faster when provided with justification**

But we couldn't test it.

Because most participants *didn't even listen* to the control condition's advice without justification.

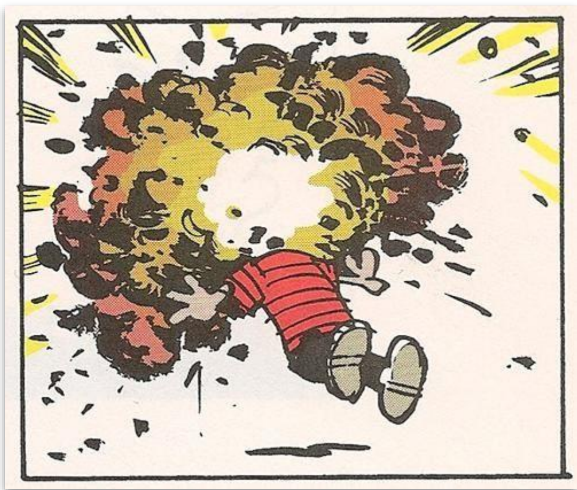


Game Completion Rate:

Control: 20%

Justification: 80%

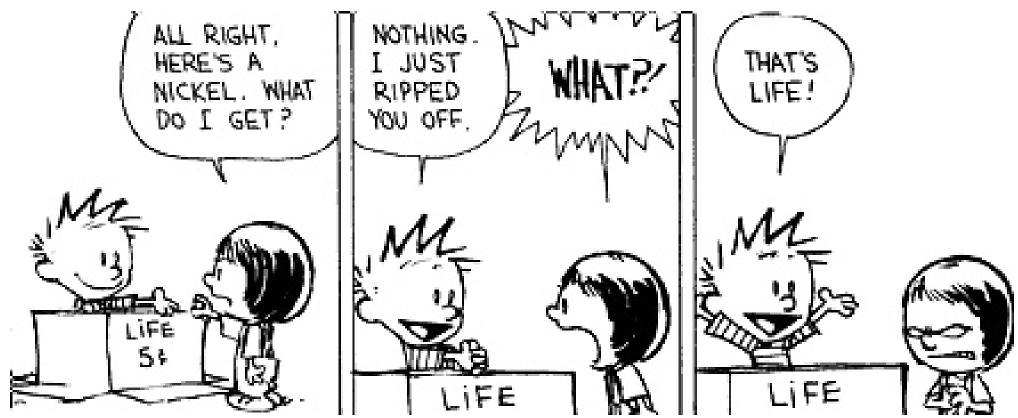
Issues and Future Work



Comprehension variables for each reward causes the state space to explode combinatorially… but rewards are rarely independent!

Justification matters…

but why?



Summary

We developed...

Reward Augmentation and Repair through Explanation

framework for using a competent agent to coach others

We evaluated...

Challenging collaborative cognitive game with a human and robot

We found...

Control condition: **Hardly anyone followed the robot's advice!**

Justification condition: **Nearly everyone followed the robot's advice!**

We showed...

RARE makes robots more useful, helpful, and intelligent coaches.

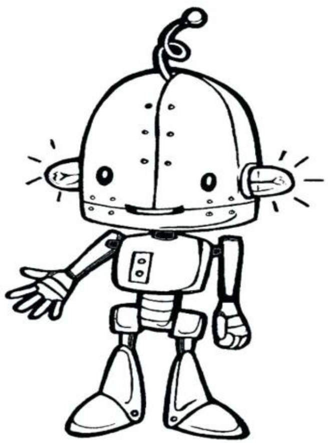
Justification is essential for effective knowledge transfer!

Control

Justification

"Sawyer wasn't forceful enough and was not giving me the reasons why the move was wrong. So I couldn't trust him"

"He was … telling me why my move was not right even though it was the right move. I was able to trust him easily when he gave the reasons"



"Response looked like hard coded and I did not find the reason to think that Sawyer was addressing to me"

"I learnt to think of moves ahead when Sawyer helped me once with the game."

"I did not believe it as it did not give details regarding the wrong step"

"Sawyer's input made me question my understanding of the game"

Skeptical of Sawyer for not giving justification

More positive user experience

Explainable AI for Human-Robot Teaming

Collaborative Artificial Intelligence and Robotics Lab

University of Colorado Boulder

Prof. Brad Hayes

Bradley.Hayes@Colorado.edu

http://www.cairo-lab.com/

@hayesbh

http://bradhayes.info