

# Robot Learning by Demonstration

Slides credit: Aalhad Patankar, Soia Chernova and Aude Billard

# Equivalent Terms in the Literature

---

*Robot Programming by Demonstration*  
*Imitation Learning in Robots*  
*Apprenticeship Learning*  
*Robot Learning from Demonstration*



# Introduction: Why learn from demonstration?

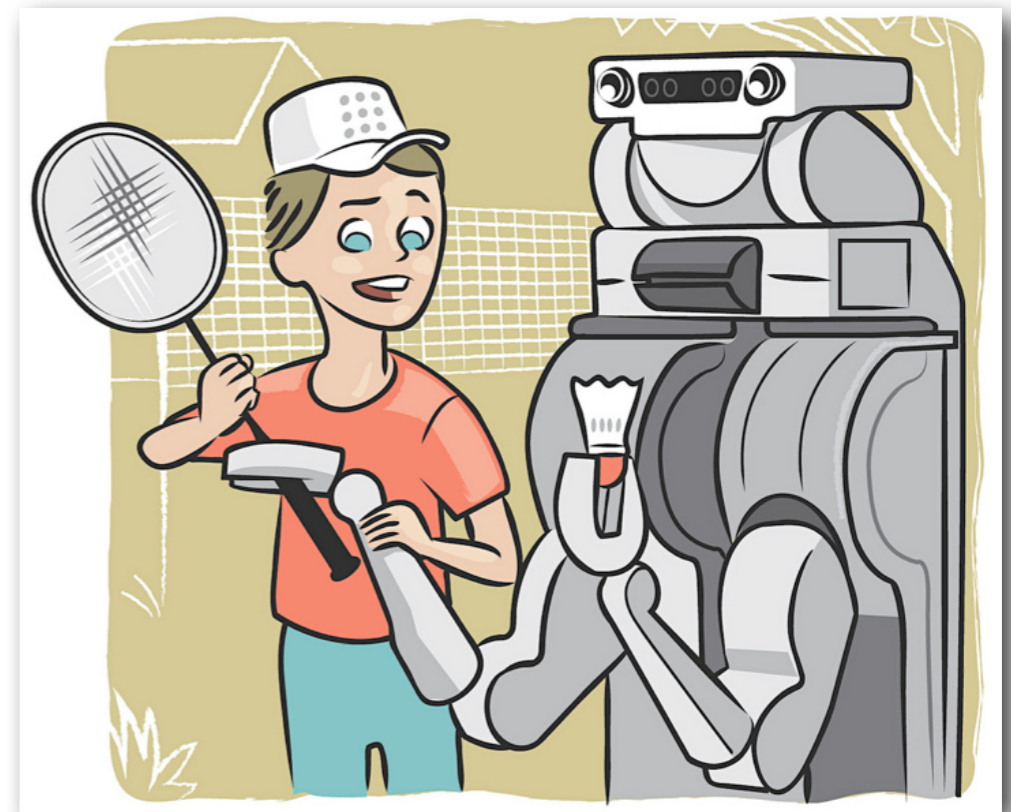
Programming robots is hard!

- Huge number of possible tasks
- Unique environmental demands
- Tasks difficult to describe formally
- Expert engineering impractical



# Introduction: Why learn from demonstration?

- Natural, expressive way to program
- No expert knowledge required
- Valuable human intuition
- Program new tasks as-needed



How can robots be shown how to perform tasks?

# Learning from Human Demonstrations: Principle

---

- Transfer to the robot skills that took years for the humans to master.
- Human can quickly re-train the robot to adapt to task changes.
- The human teaches by showing how to perform the task.



# Overview of learning from demonstration (LfD)

- Learning from Demonstration: Deriving a policy from *examples* provided by a teacher
- Different from reinforcement learning, in which a policy is derived from *experience*, such as exploration of different states and actions in reinforcement learning

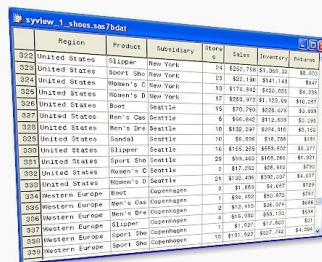
# What is learning from demonstration (LfD)?

- Policy: a mapping between actions and world state
  - E.g. moving an actuator (action) and the location of a box near the robot (world state)
- Examples: A sequence of state-action pairs that are recorded by some sort of teacher demonstrator



Teacher

demonstration

A screenshot of a data table with columns: Region, Product, Subsidiary, Store, Sales, Priority, Interest. The table contains 20 rows of data, including entries for 'United States', 'Western Europe', and 'Northern Europe' with various products like 'Slider', 'Sport Shoe', and 'Women's C'.

Policy derivation



Learner



# Two phases of LfD

- Gathering examples: the process of recording example data to derive a policy from
- Deriving policies: analyzing examples to determine a policy

# Advantages of LfD

- Does not require expert knowledge of domain dynamics, which depends heavily on the accuracy of the world model
- Intuitive, as humans already communicate knowledge in this way
- Demonstration focuses the dataset only to area in the state-space encountered during demonstration

# Formal definition

- World consists of *states*  $S$  and *actions*  $A$
- States  $Z$  are *observable states* which are mapped from  $S$  to  $Z$  by mapping  $M$
- A *policy*  $\pi : Z \rightarrow A$  is a selection of actions  $A$  based on the observable world states

# Design choices: demonstrator

- Choice of demonstrators have big impacts on the algorithms used for derivation of policy
- Can be broken down into who designs the demonstration, and which body executes the demonstration
  - E.g. human designer tele-operating a robot, robot designing and executing demonstration
- Human demonstrators usually used

# Design choices: demonstration technique

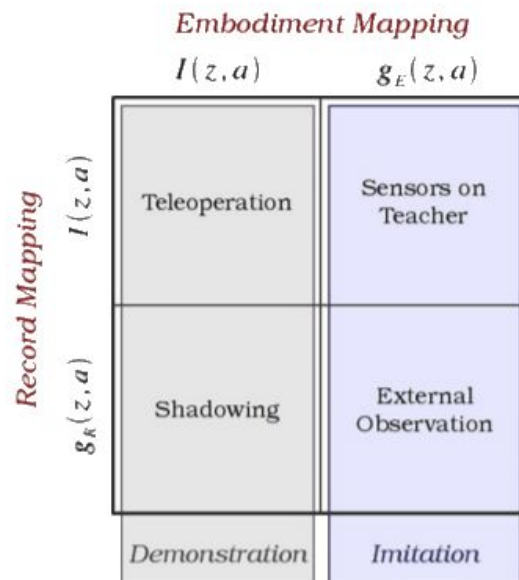
- Whether policy is derived *after* all training data is obtained (batch), or is developed incrementally as data becomes available (interactive)
- Problem space continuity: whether states are discretized or continuous
  - Discretized example: states broken as “box on table,” “box held by robot,” “box on floor” etc
  - Continuous example: in same example, using 3D position of robot’s effectors and box throughout actions
- Continuity of problem space has big effects on what algorithms are used in the policy derivation stage

# Building the example dataset: correspondence

- Because of differences in the teacher's sensors and actuators (human eyes, human joints) and the robot's sensors and actuators, a direct transfer of information from teacher to student is often difficult
- This issue, called **correspondence**, and can be broken down into two categories:
  - Record mapping: correspondence between teacher's actions and recorded data
  - Embodiment mapping: correspondence between recorded data and learner's execution

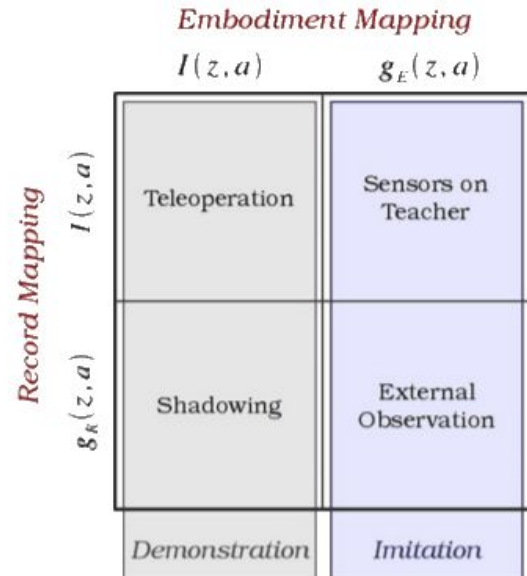
# Building the example dataset: correspondence

- Data acquisition for LfD can be broken down into categories based on correspondence
- $I(z,a)$  means identity function (direct mapping), while  $g(z,a)$  is a mapping function used for correspondence



# Teleoperation

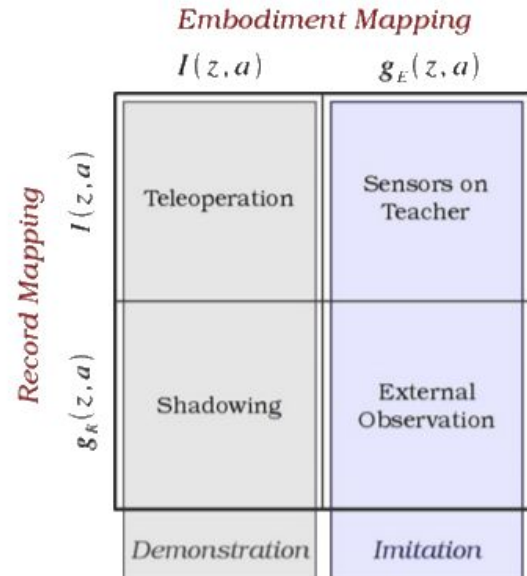
- Human operator controls a robot teacher
- Direct record and embodiment mapping, as all recording and execution is done on the student body itself by human operator
- E.g. human controlling a robot's movements through remote control to teach it to find a box





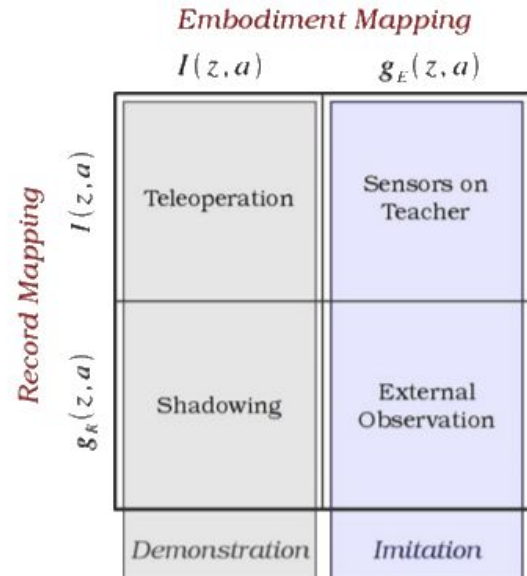
# Shadowing

- Robotic platform shadows human teacher, and recordings are done from robotic platform
- Direct embodiment mapping because robot's own sensors are used to record data, but record mapping required between human actions and robot demonstration in shadowing step



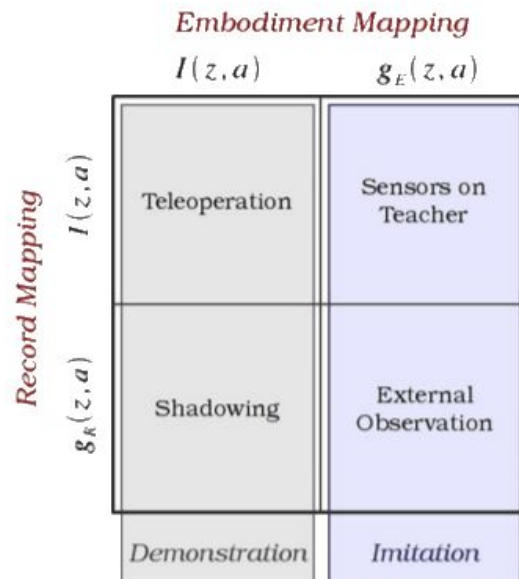
# Sensors on Teacher

- Sensors are placed directly on teaching platform, so record correspondence issues are alleviated
- Can come with large overhead such as specialized sensors and a customized environment



# External observation

- Sensors external to the body executing the demonstration are used to record data
- Less reliable and less precise, but comes with less overhead



# Deriving a policy: mapping function

- Attempts to calculate the underlying function behind the states and actions and generalize over set of training data
- Two major categories: *classification* and *regression*
- Is heavily influenced by demonstration design choices mentioned earlier

# Mapping function: Categorization

- Input is categorized into discrete classes and outputs discrete robot actions
- Many algorithms, such as k-Nearest Neighbors, Gaussian Mixture Models, and Bayesian networks are used to perform the classification, depending on the application
- Can be done for low level robot movement (controlling a car in a simulated environment), mid-level motion primitives (teaching a robot to flip an egg), and high level complex actions (ball sorting task)

# Mapping function: Regression

- Maps demonstration states to continuous action outputs
- Lazy learning: function approximation is done “on demand” whenever a current observation needs to be mapped at run-time
- At opposite end, all function approximation done prior to run-time
  - No adjustments to policy done at run-time
  - Very computationally expensive

# Plans

- Actions are composed of *pre-conditions*, the state that must take place before an action can occur , and *post-conditions*, the state immediately after the action
- Non state-action information, such as *intentions* and *annotations* can be provided by the teacher to the learner in addition to demonstration data

# Example with plans: clearing a table

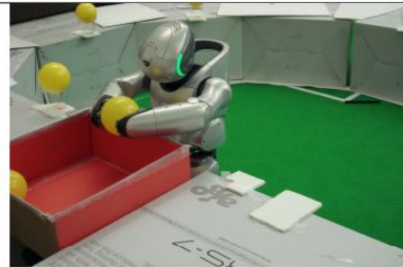
- Task: clearing a table
- Pre-programmed actions: pick, drop, search, etc. available to robot
- After demonstration, robot learns how these actions relate to objects and states, and learns mapping between sequence of actions and states



Pick Object



Carry Object



Drop Object

Veeraragha H, Veloso M.  
“Teaching Sequential Tasks  
with Repetition through  
Demonstration.” International  
Foundation for Autonomous  
Agents and Multiagent  
Systems, 2008.



# Overview of current research areas

## Low-Level Skills

- Trajectories
- Force profiles

## High-Level Skills

- Combination of actions
- Speech-directed teaching

## Batch learning versus incremental learning

## Combined with other techniques

- Bootstrap reinforcement learning
- Inverse optimal control

## User-studies to assess:

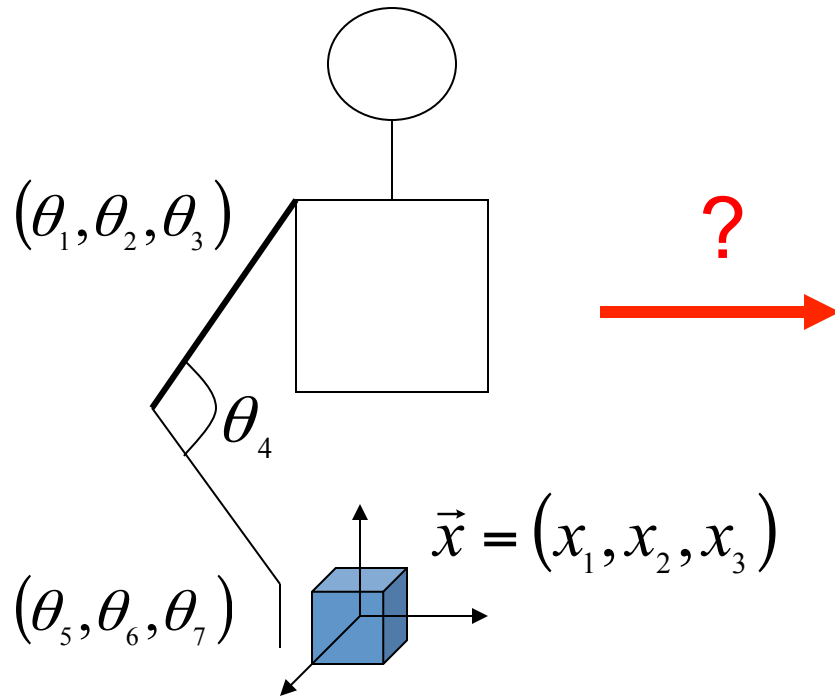
- Interfaces
- Effectiveness of algorithm

# Not Just Record and Replay: Generalize!

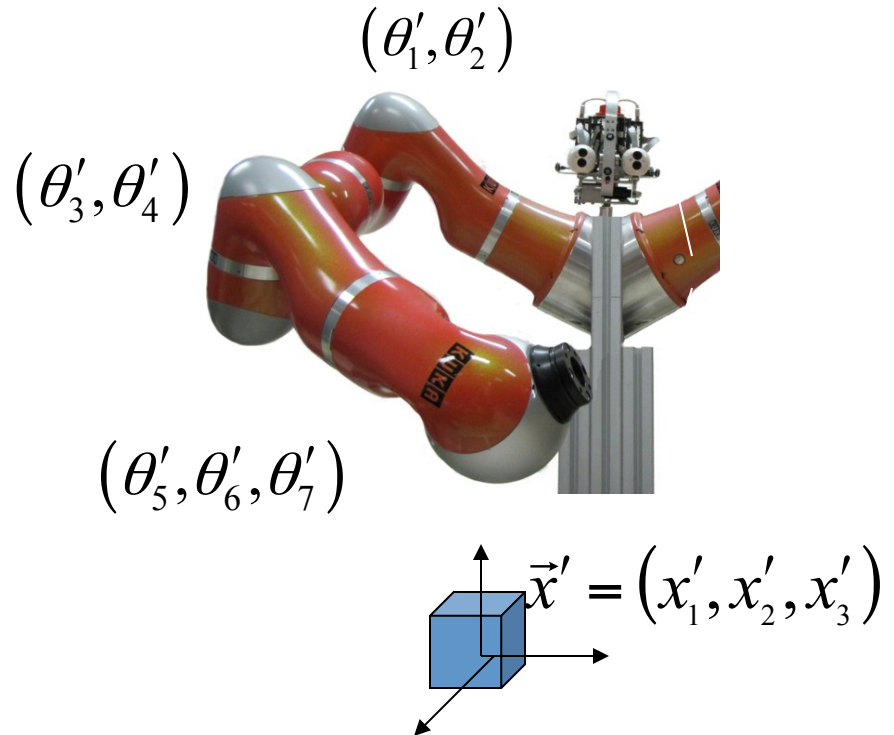


# Correspondence Problem

Demonstrator

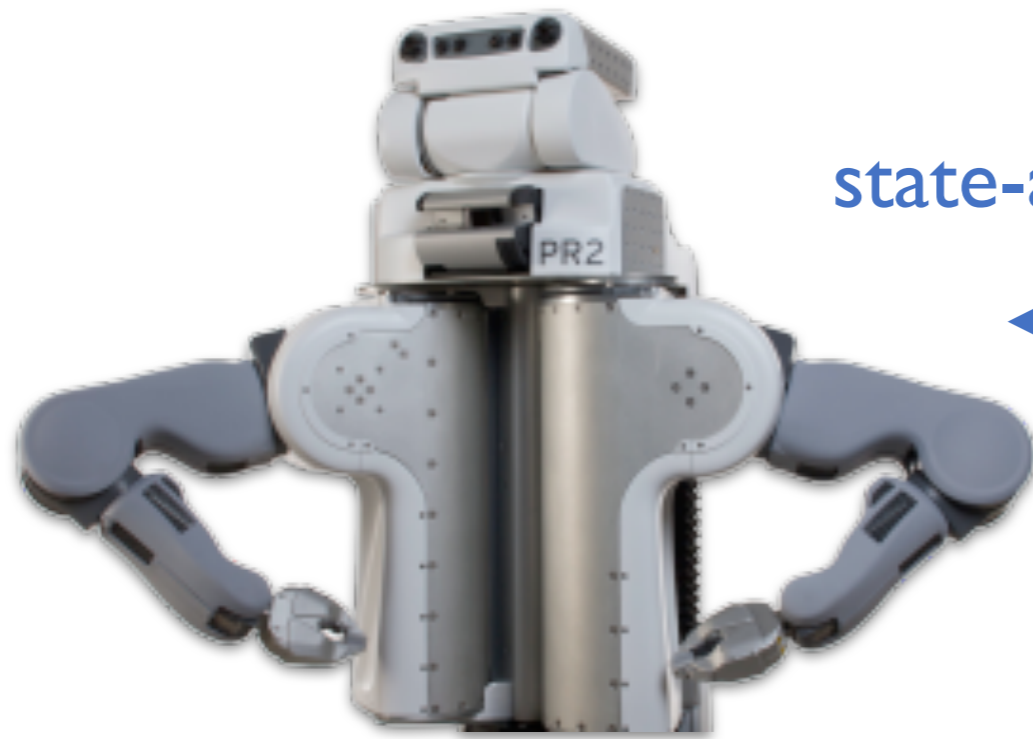


Imitator

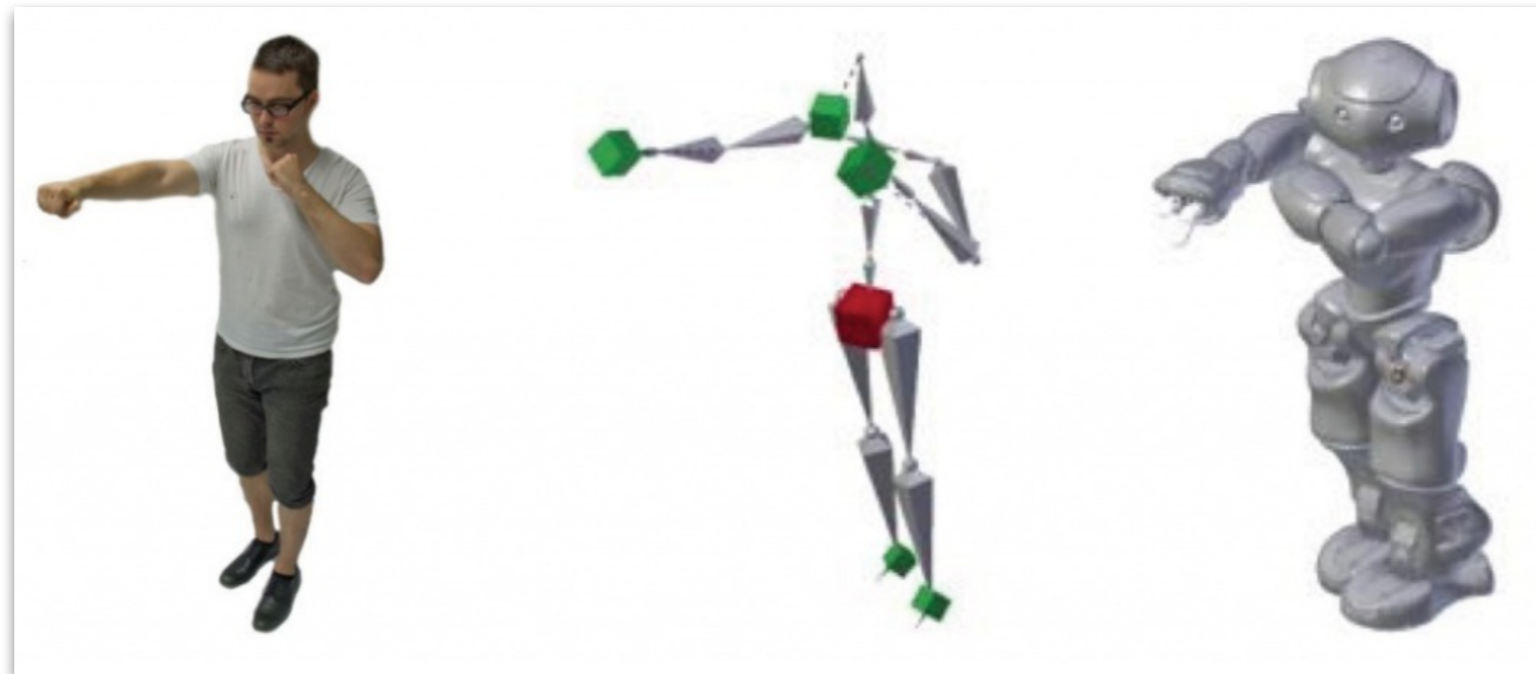


Establish a correspondence across degrees of freedom when feasible.

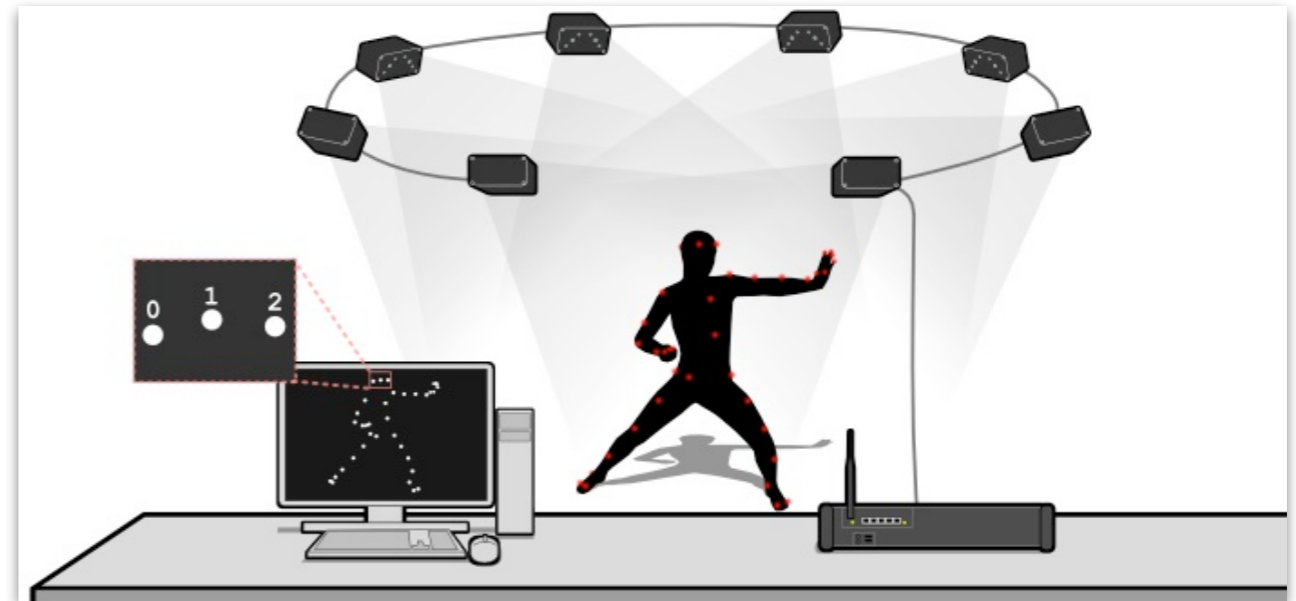
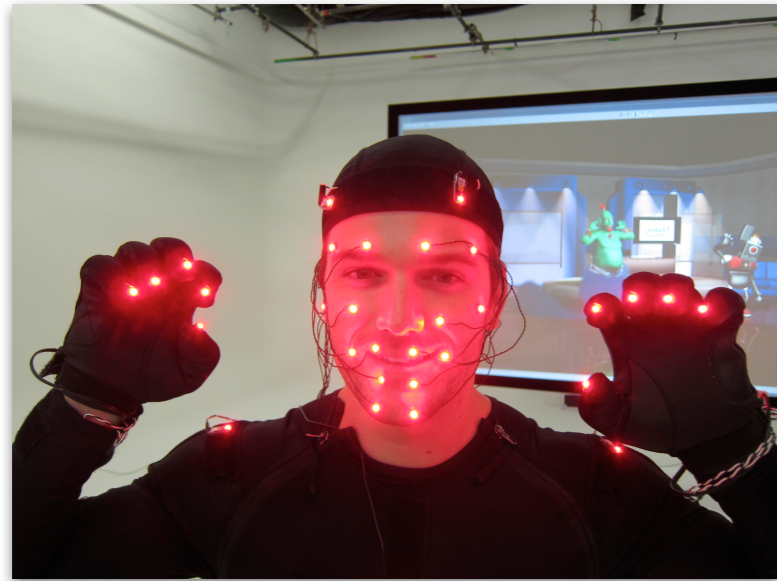
# The correspondence problem



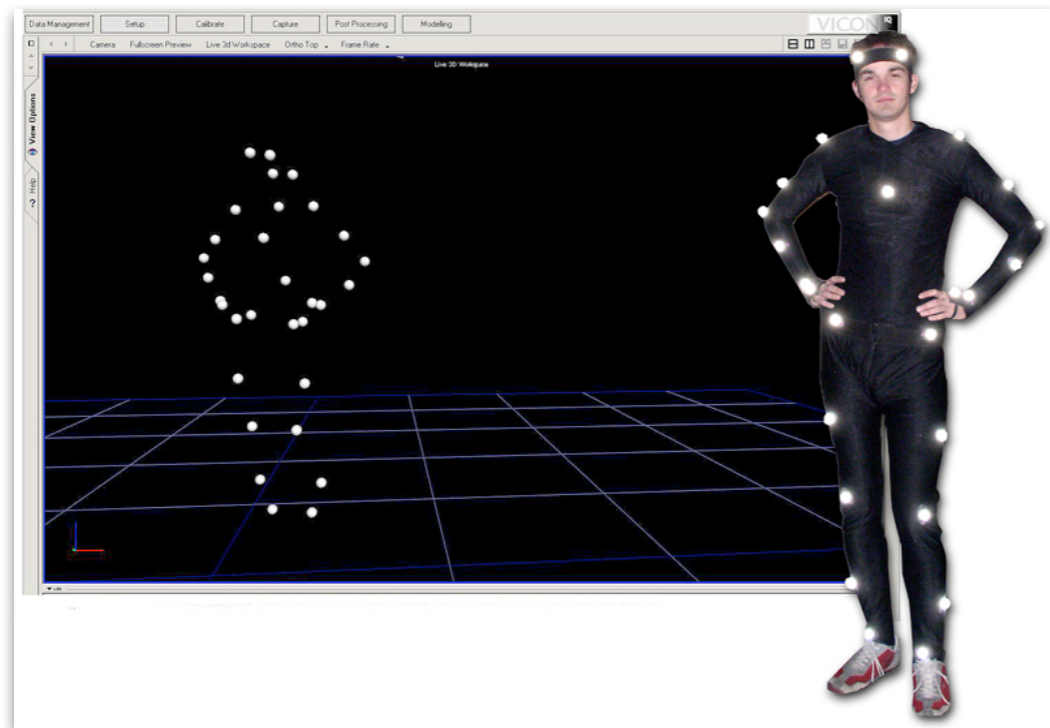
state-action mapping?



# Sensing: Motion capture

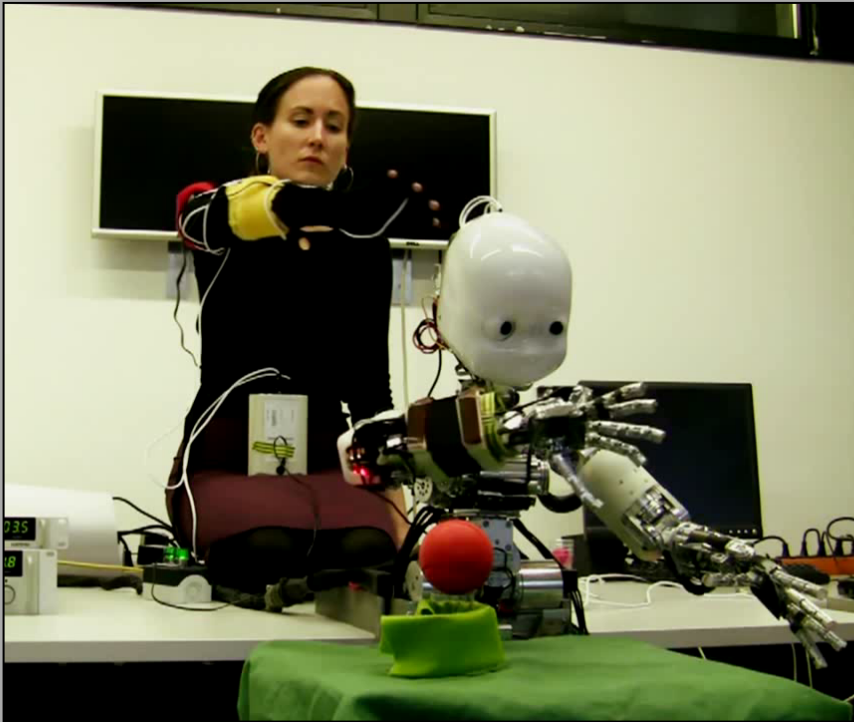


Phasespace



Vicon

# Which interface?



## Motion sensors:

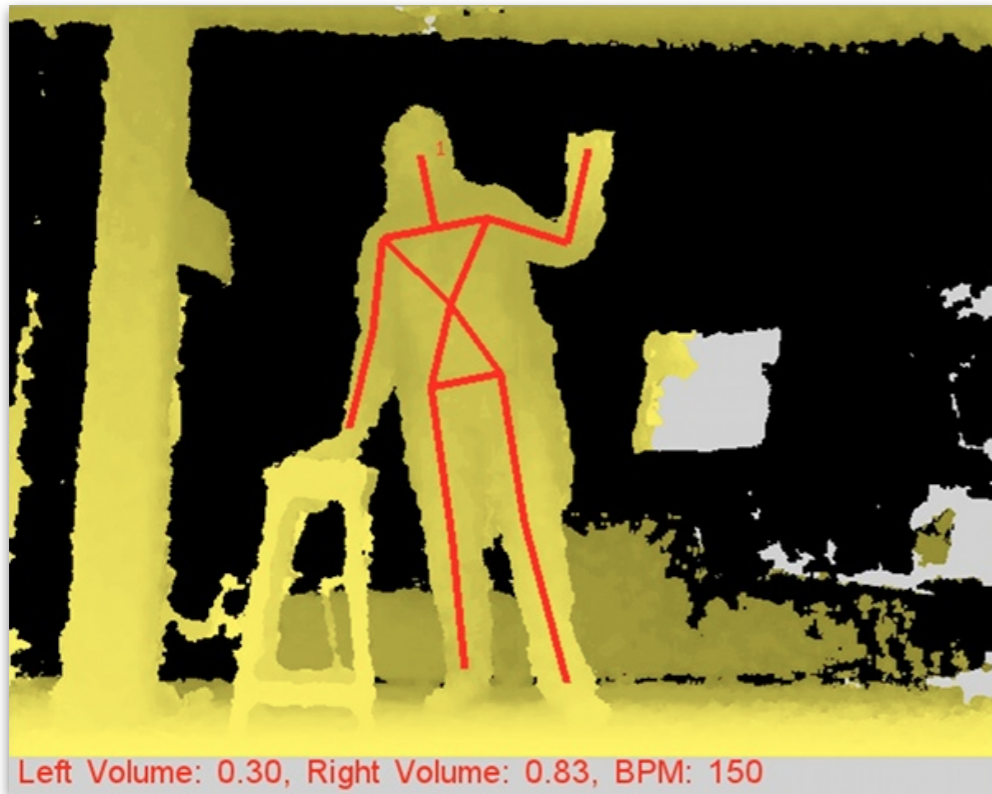
### Pros:

- Real-time kinematic information
- Solve correspondence problem

### Cons:

- Require to wear the system
- No haptic information

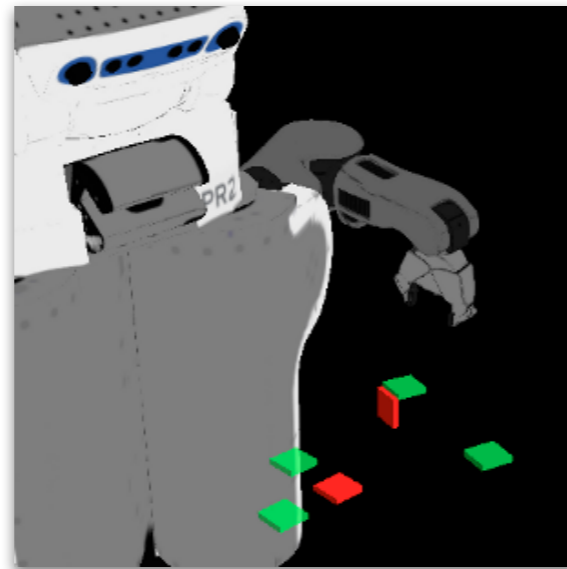
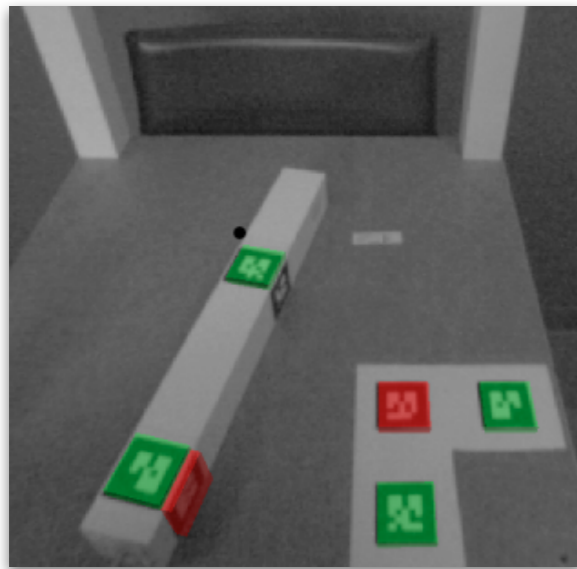
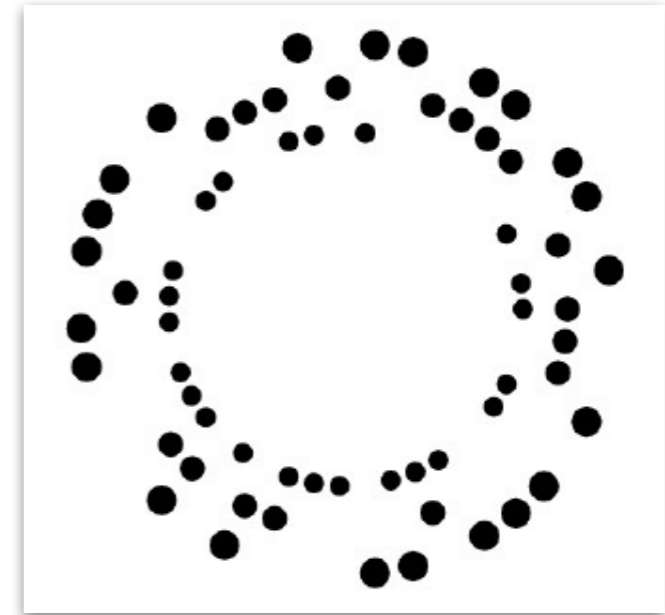
# Sensing: RGB(D) cameras, depth sensors



- Standard RGB cameras
- Stereo: Bumblebee
- **RGB-D: Microsoft Kinect**
- Time of flight: Swiss Ranger
- LIDAR: SICK

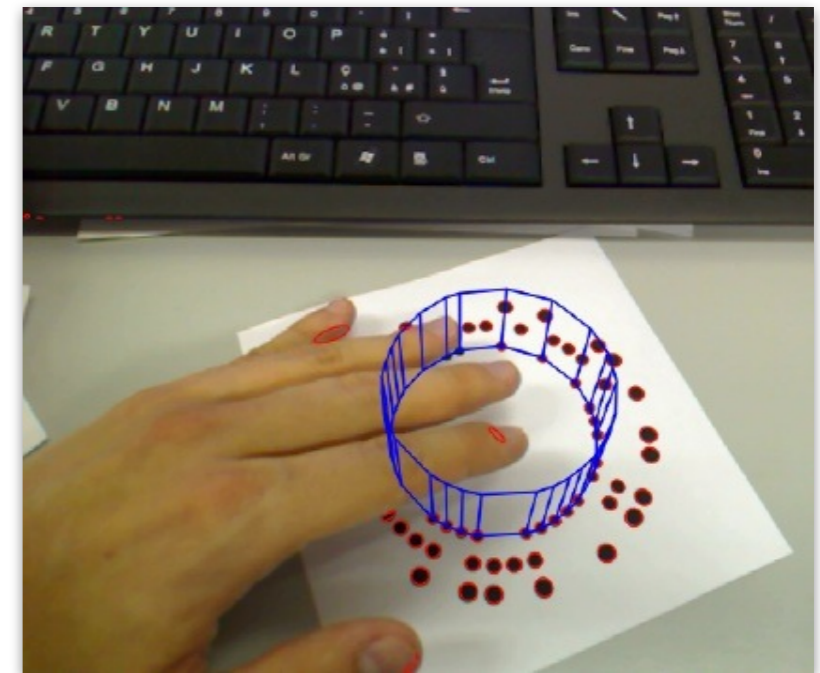


# Sensing: Visual fiducials



AR tags

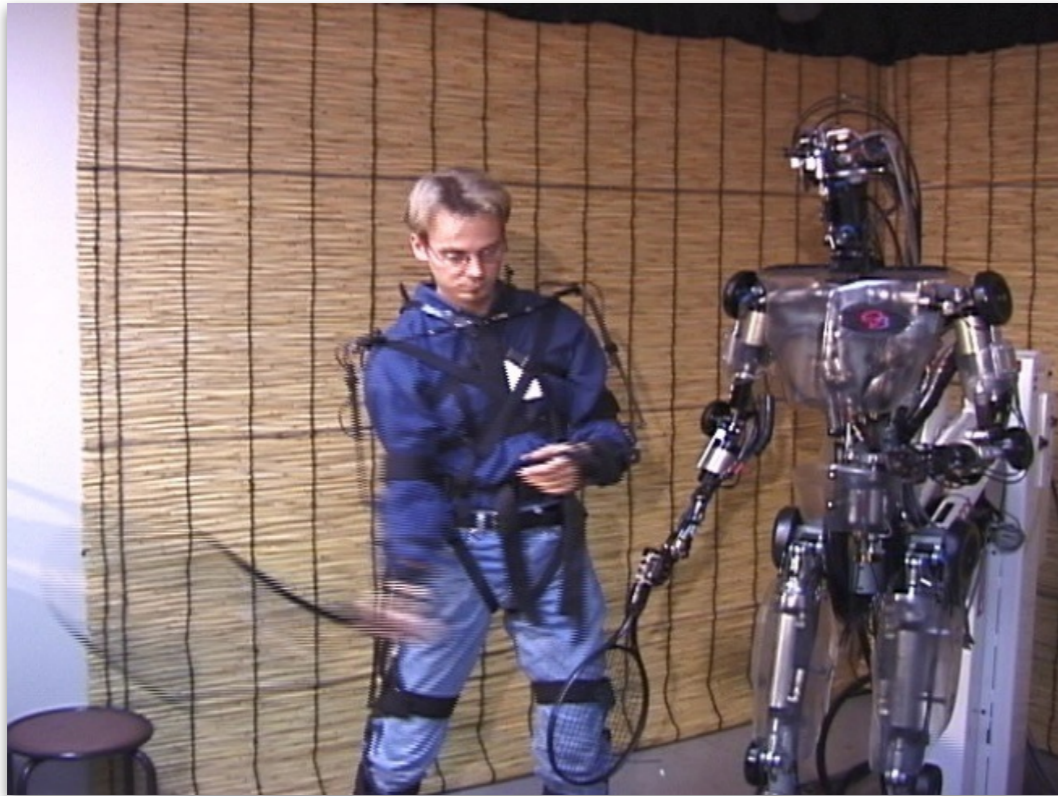
[http://wiki.ros.org/ar\\_track\\_alvar](http://wiki.ros.org/ar_track_alvar)



RUNE-129 tags

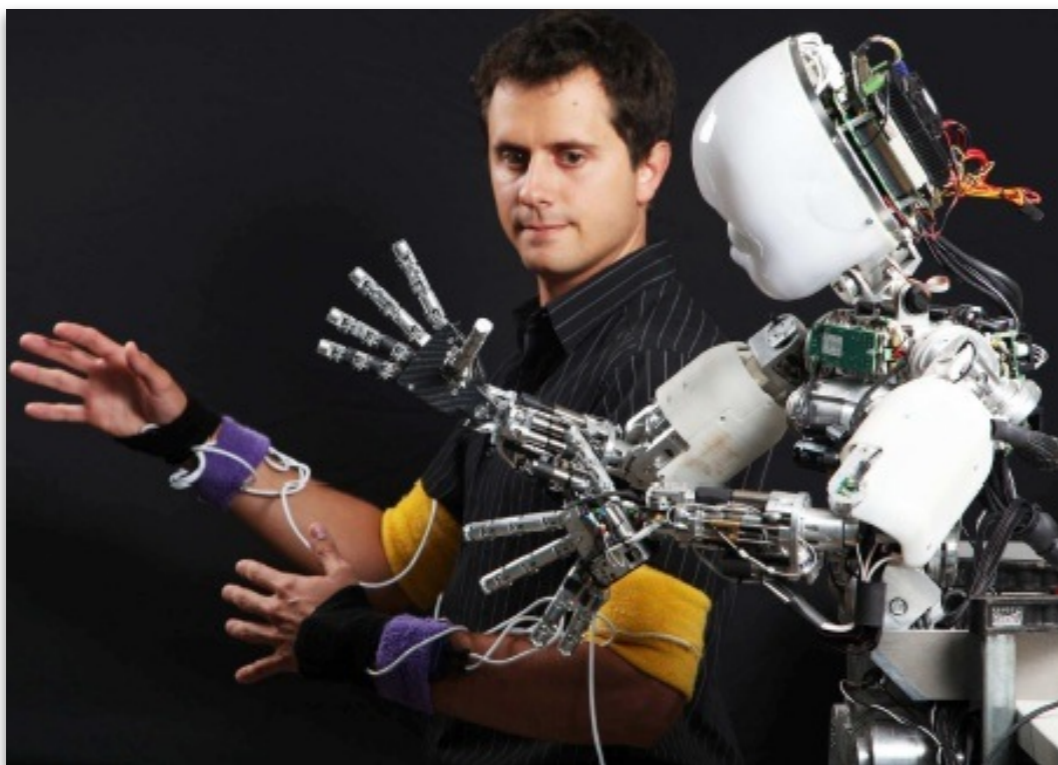


# Sensing: Wearable sensors



SARCOS Sensuit:

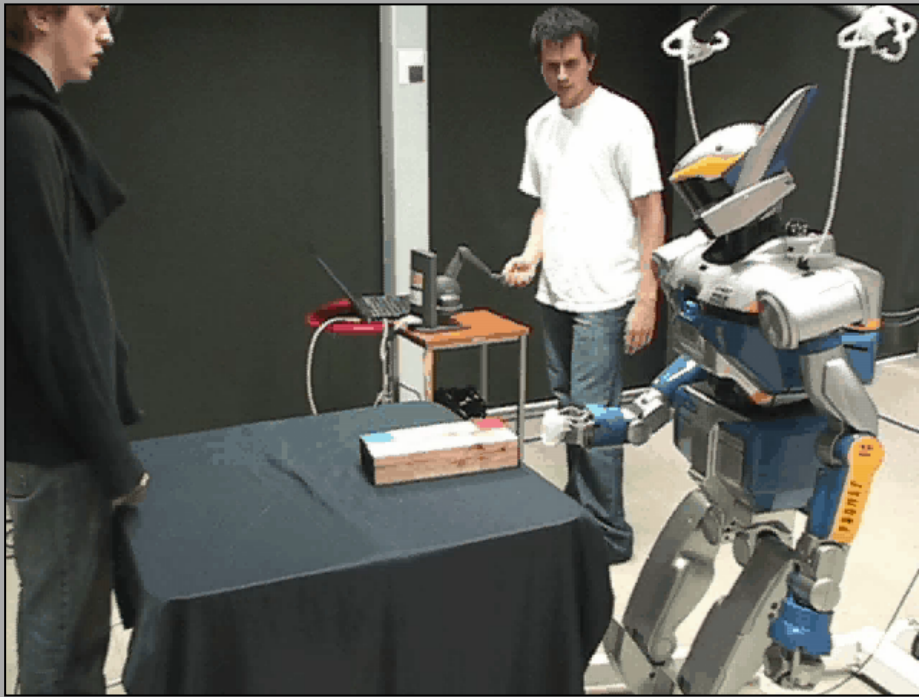
Record 35-DOF poses  
at 100 Hz



Other wearables:

- Accelerometers
- Pressure sensors
- First-person video

# Which interface?



## Haptic devices:

### Pros:

- Solve correspondence problem
- Transmit kinematic & haptic information

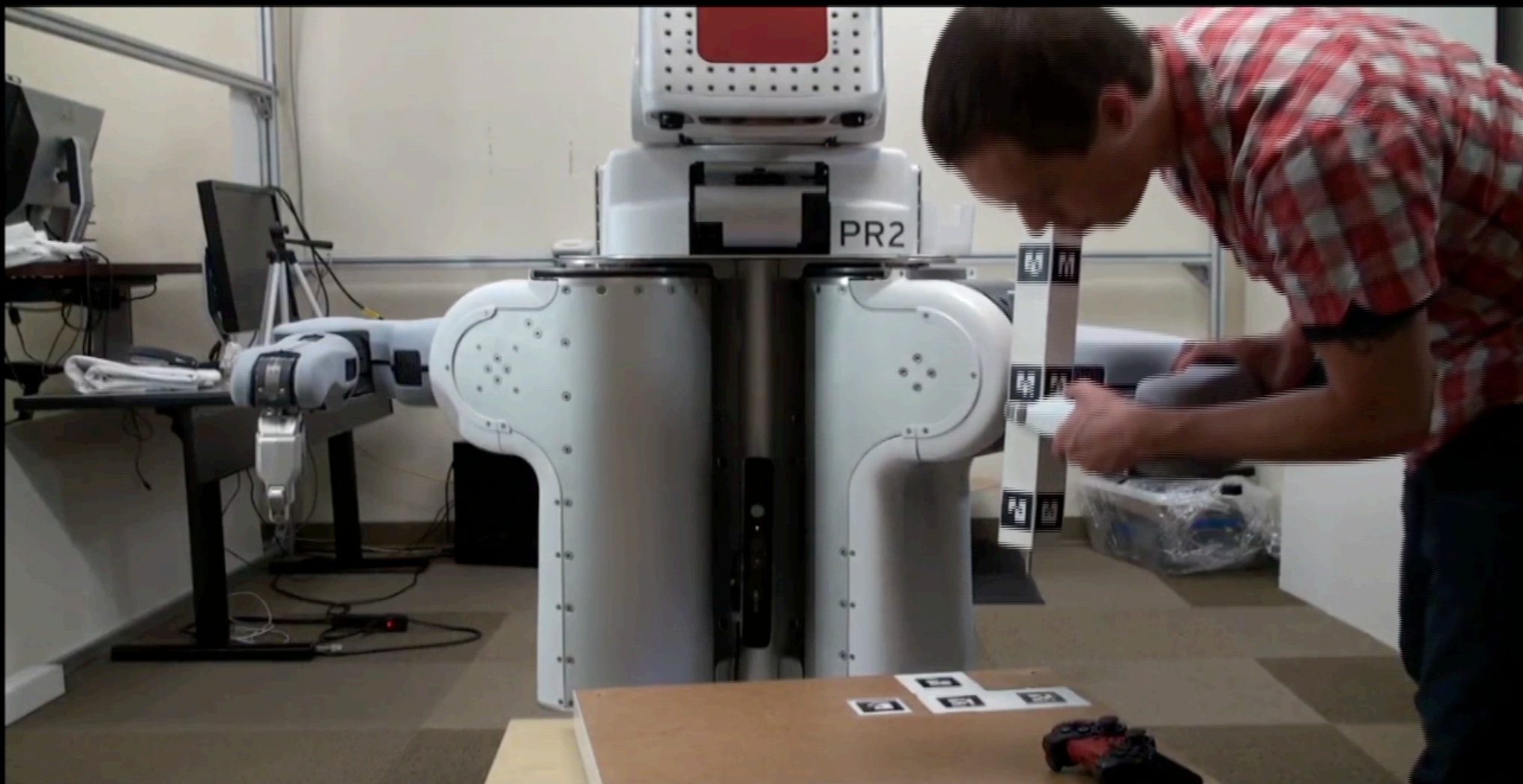
### Cons:

- Requires training
- User far from task location

# Learning by doing: Teleoperation

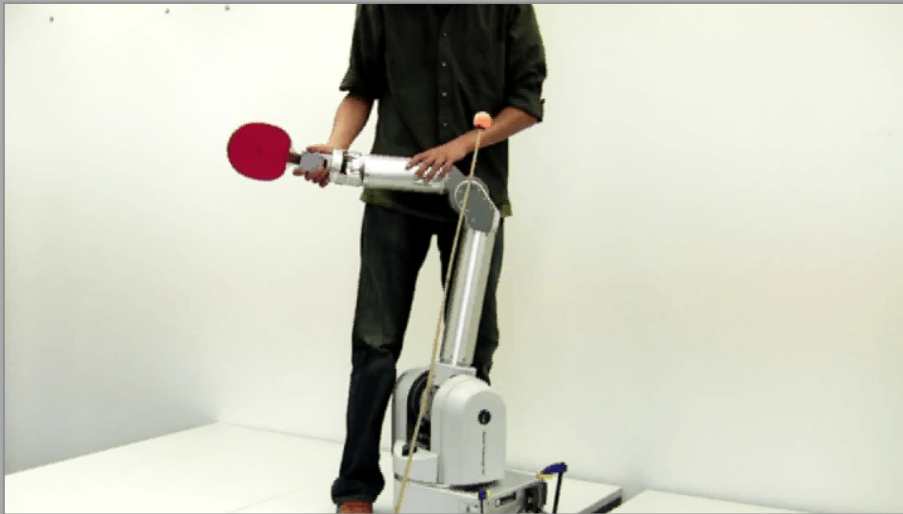


# Learning by doing: Kinesthetic demonstration



4x

# Which interface?



## Kinesthetic Teaching:

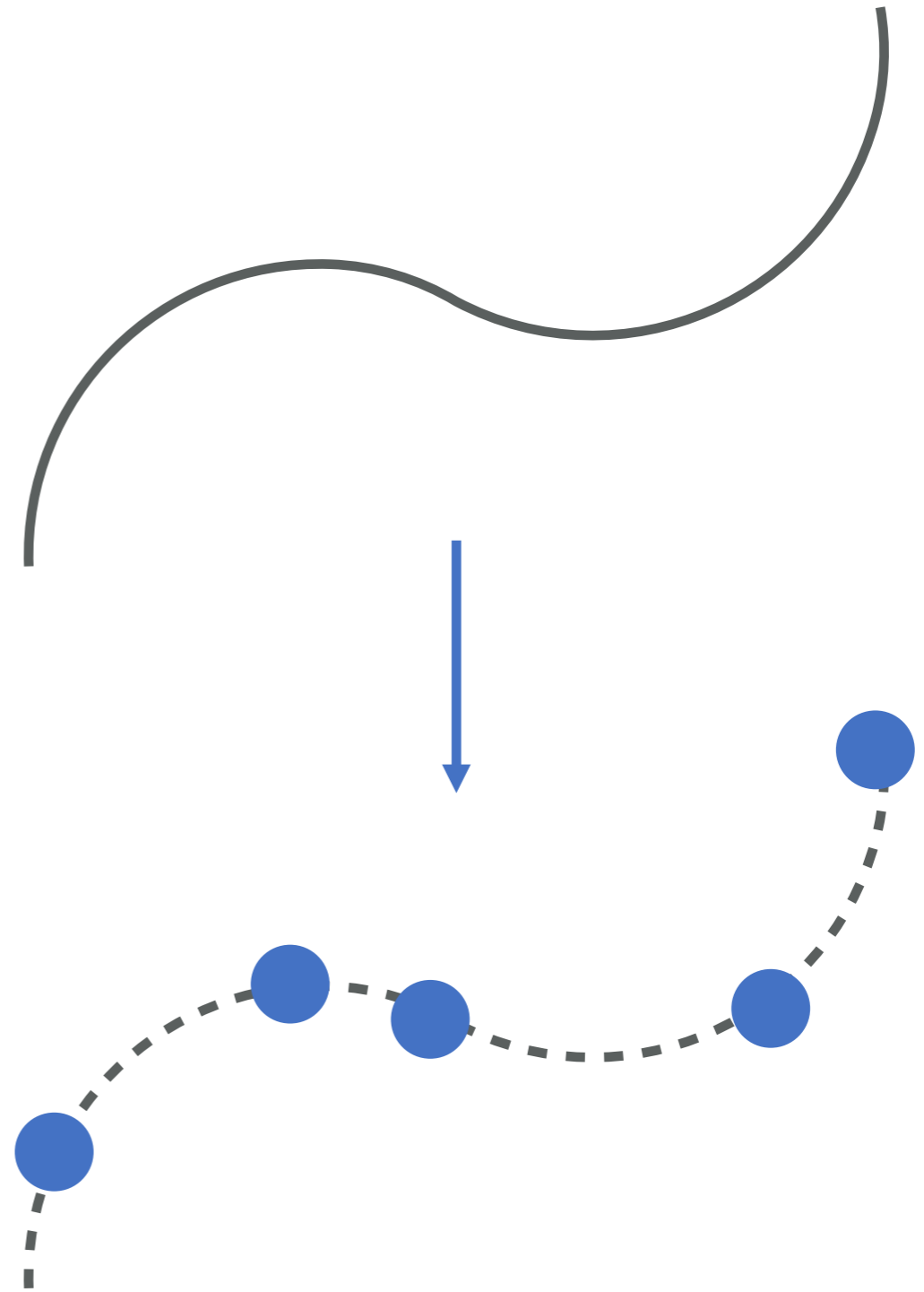
### Pros:

- Solve correspondence problem
- Transmit kinematic & haptic information

### Cons:

- Need two hands to teach movements of a few DOFs

# Learning by doing: Keyframe demonstration



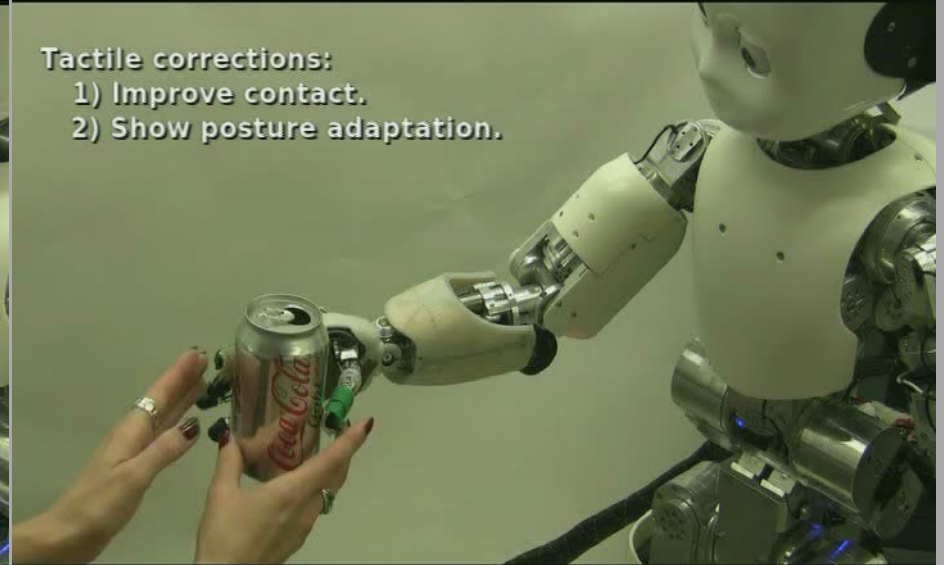
# Kinesthetic Teaching using Tactile Sensing

By contrast, when no model is used, the posture does not adapt.



*No Adaptation*

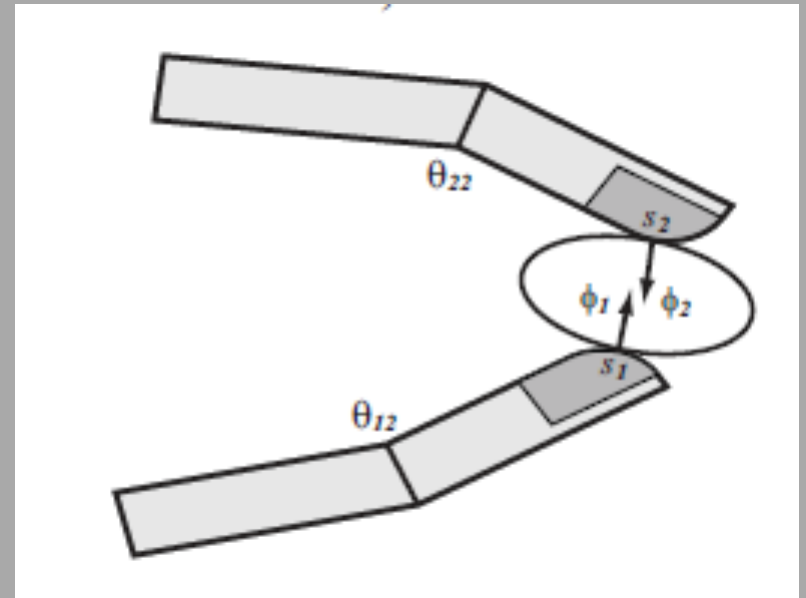
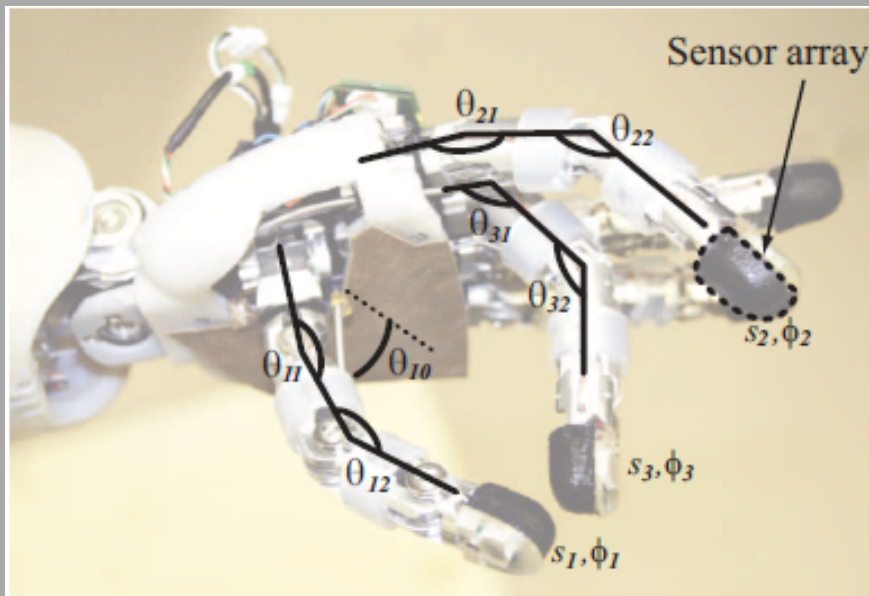
Tactile corrections:  
1) Improve contact.  
2) Show posture adaptation.



*Teaching adaptive behavior*

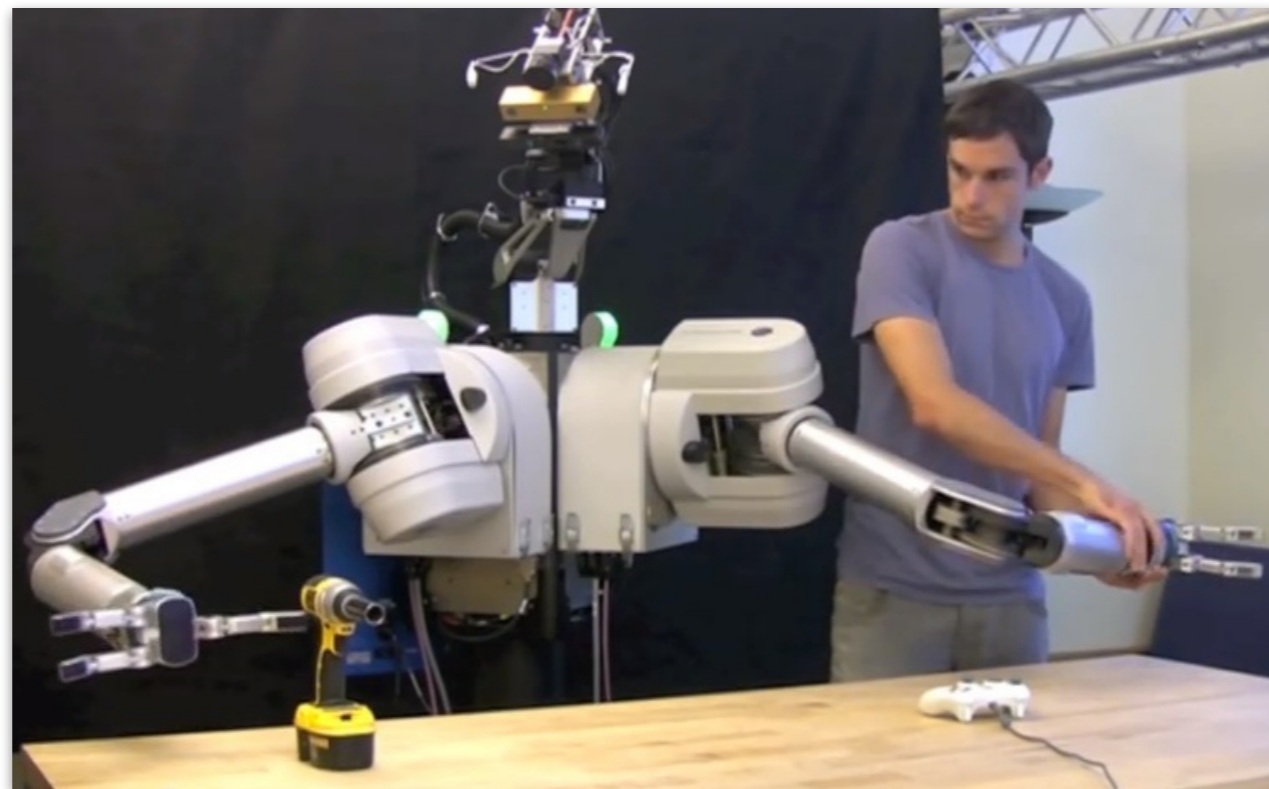
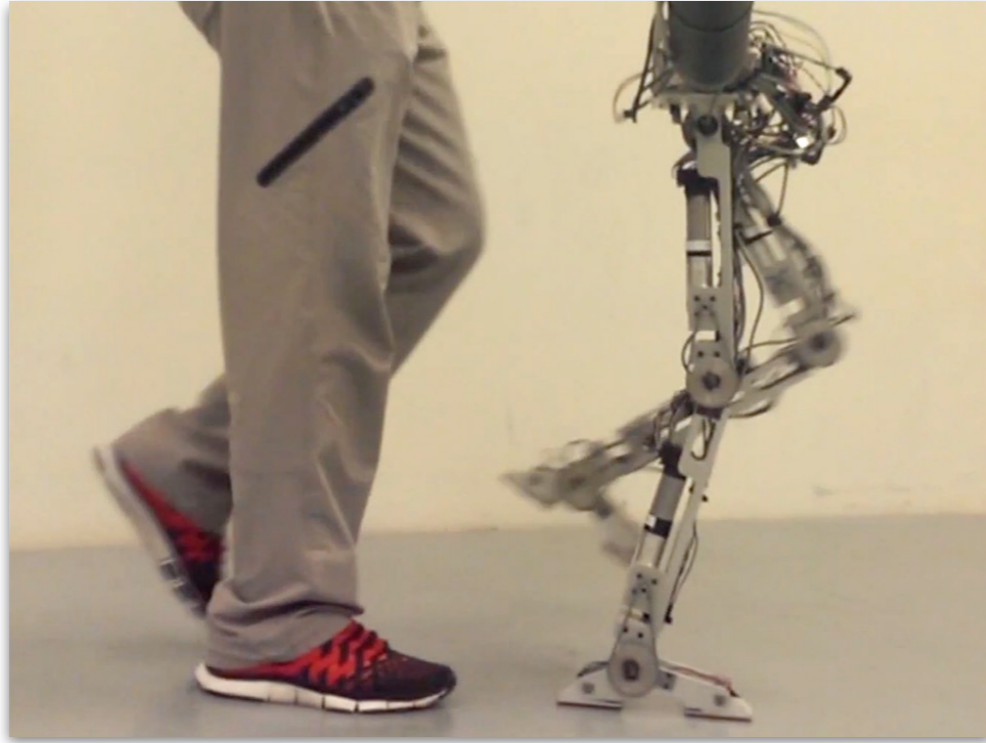
# Kinesthetic Teaching using Tactile Sensing

Learn a probabilistic mapping  $p(\phi, s, \theta)$  between contact signature of the object (normal force  $\phi$  and tactile response  $s$ ) and fingers' posture  $\theta$ .

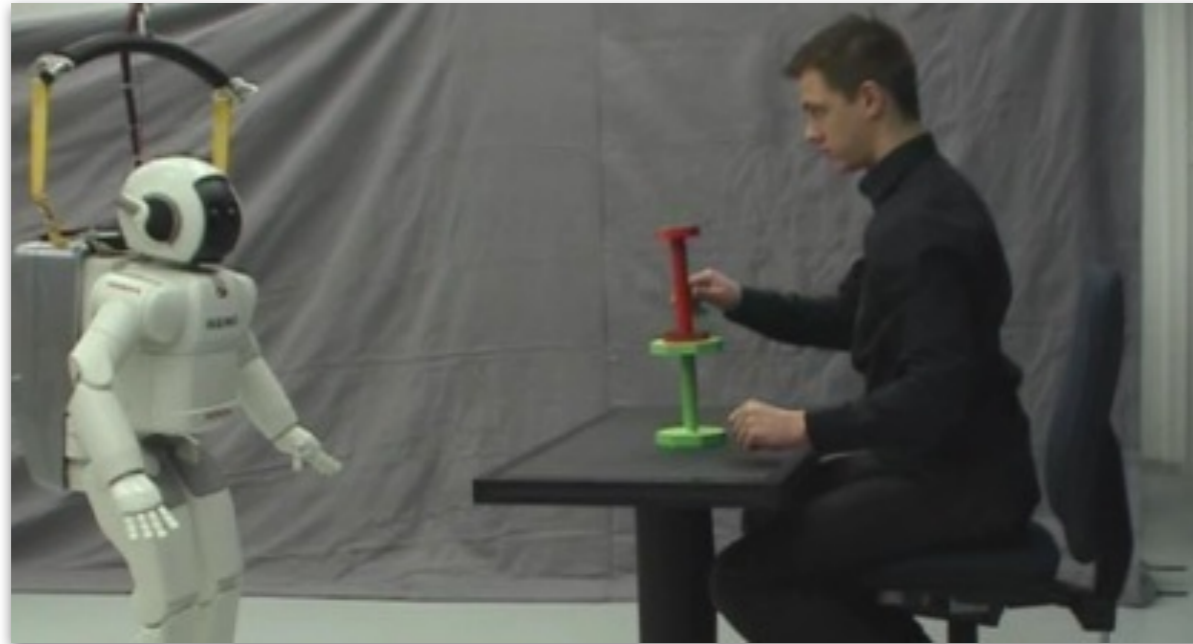




# Learning by watching: Shadowing



# Learning by watching: Simplified mimicry

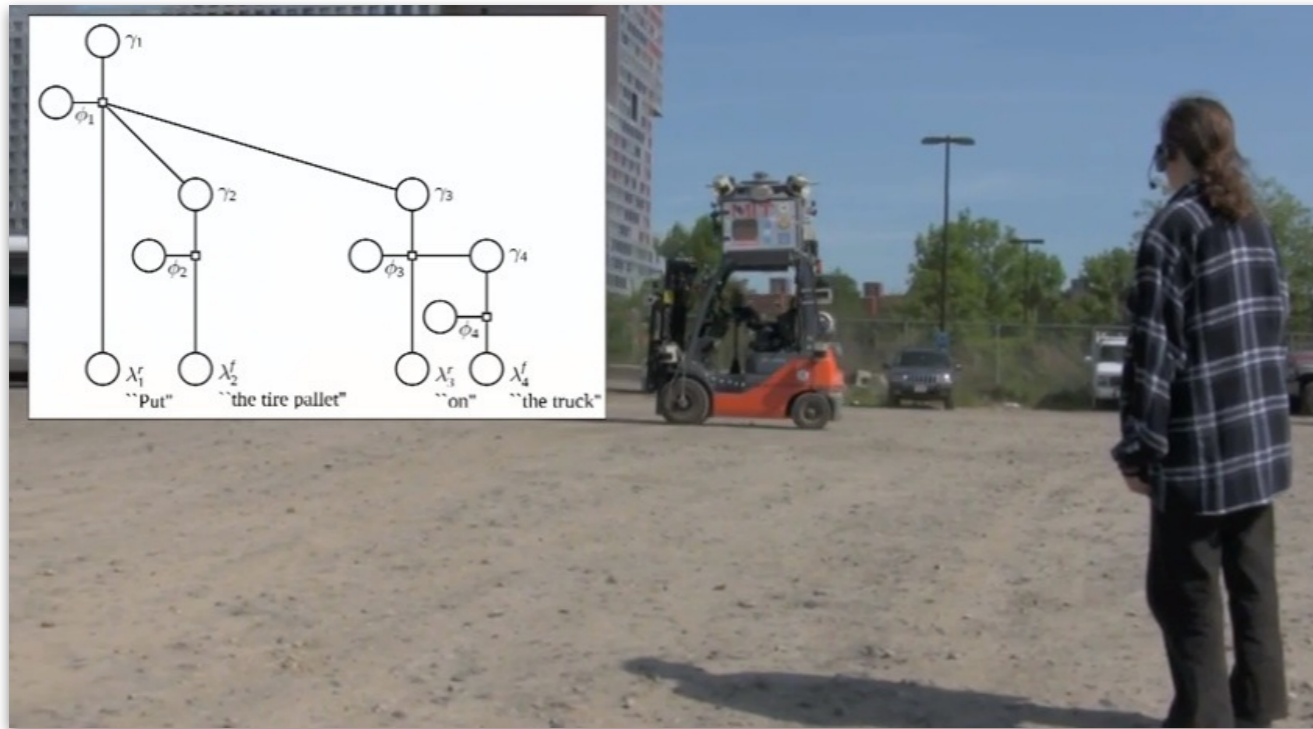


Object-based



End effector-based

# Supplementary information: Speech and critique



Interpreting and grounding natural language commands

**TAMER**  
training the robot to KEEP CONVERSATIONAL DISTANCE

00:10:01:10  
ELAPSED TIME

**LARG** Learning Agents Research Group  
**IRG** Personal Robots Group

INCOMING REWARD

REWARD KEY  
 $\leq -5$  0  $\geq 5$

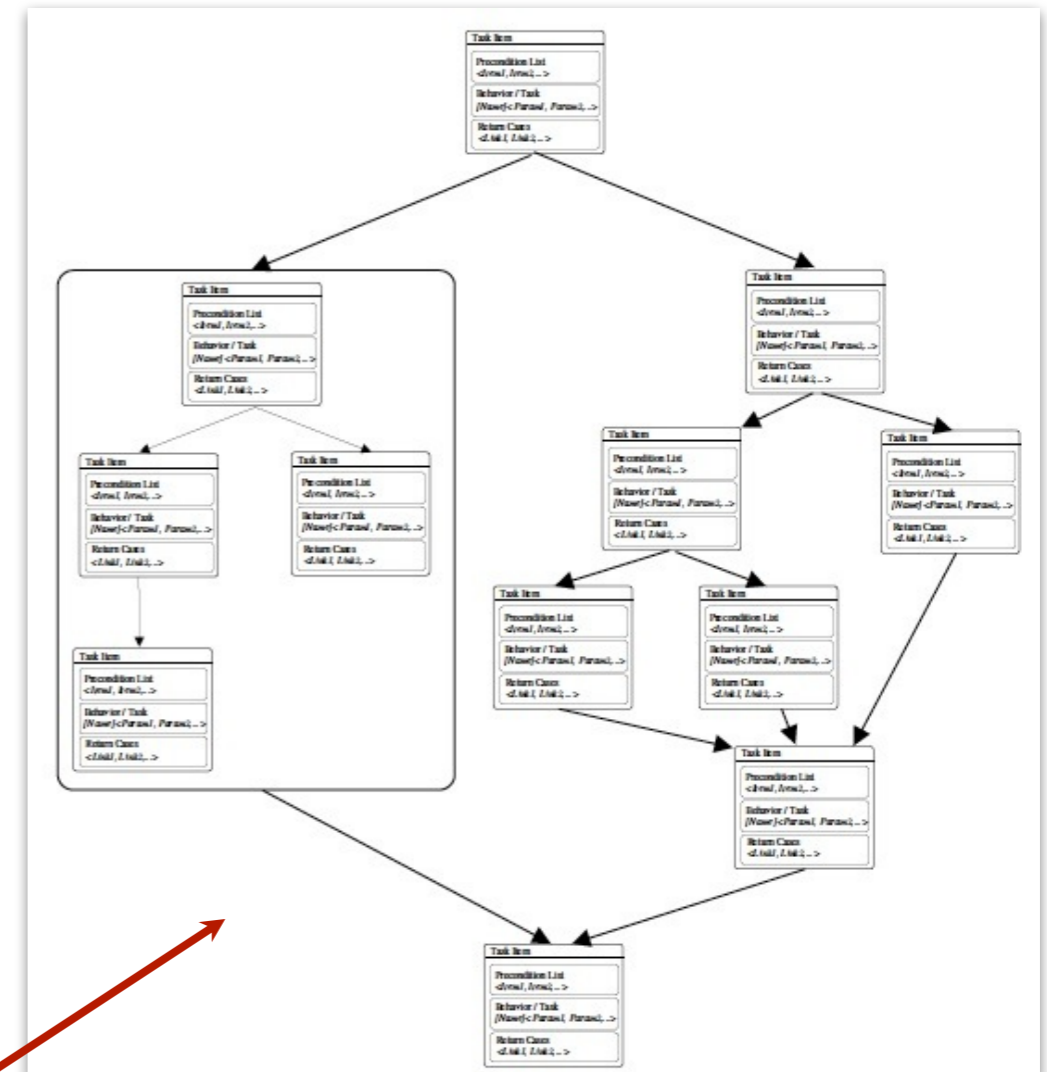
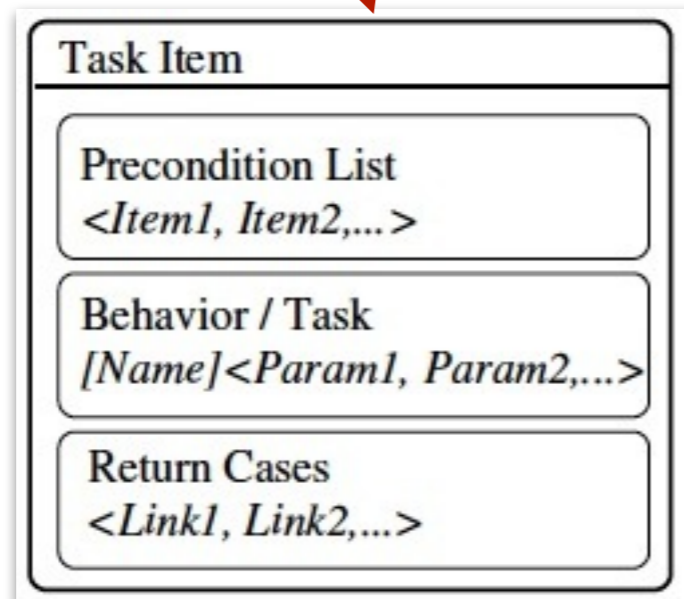
REWARD PREDICTIONS

STAY GO FORWARD TURN LEFT TURN RIGHT

Realtime user feedback given to RL system

# Learning a task plan: STRIPS-style plans

```
>> When I say deliver message
>> If Person1 is present
>>   Give message to Person1
>> Otherwise
>>   If Person2 is present
>>     Give message to Person2
>>   Otherwise
>>     Report message delivery failure
>>   Before
>> Before
>> Goto home
```



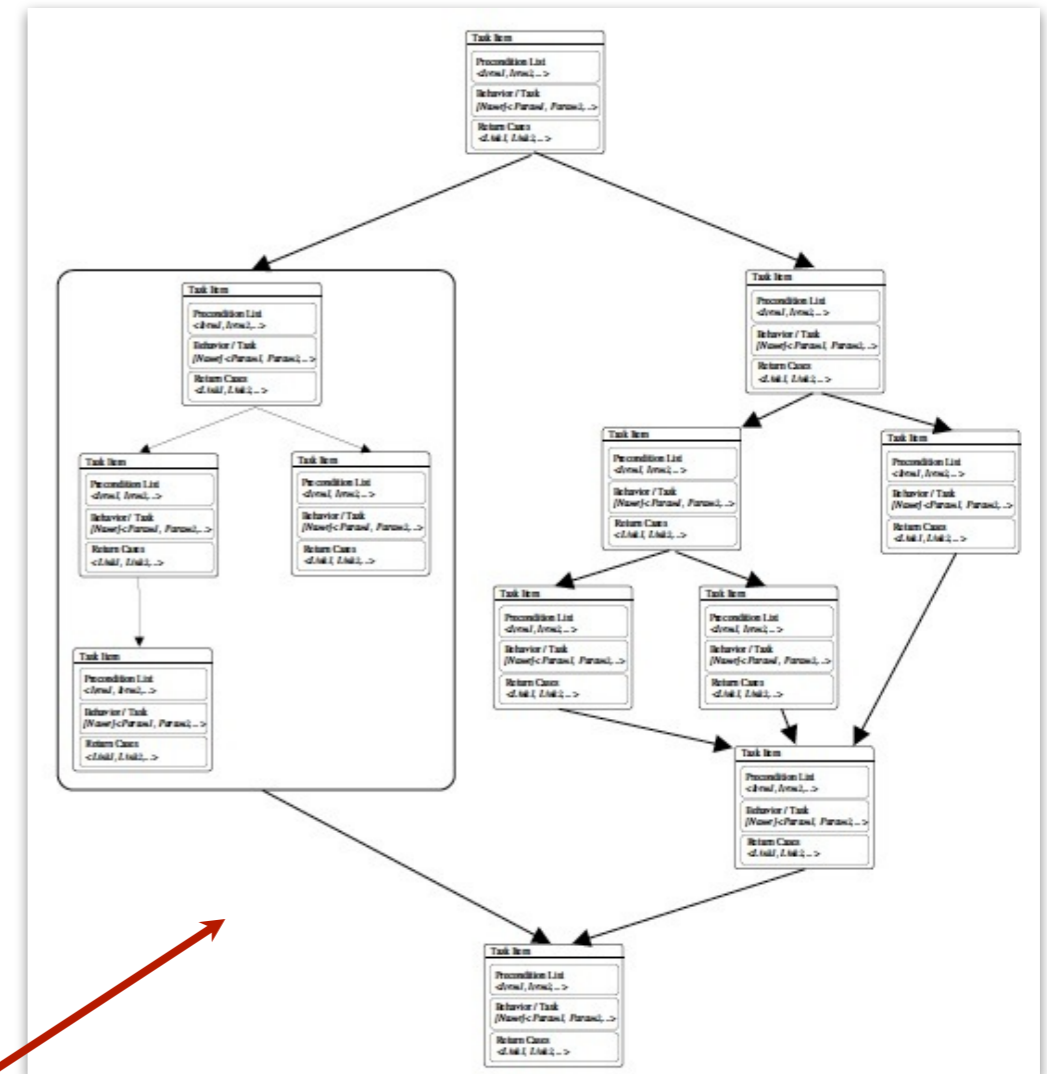
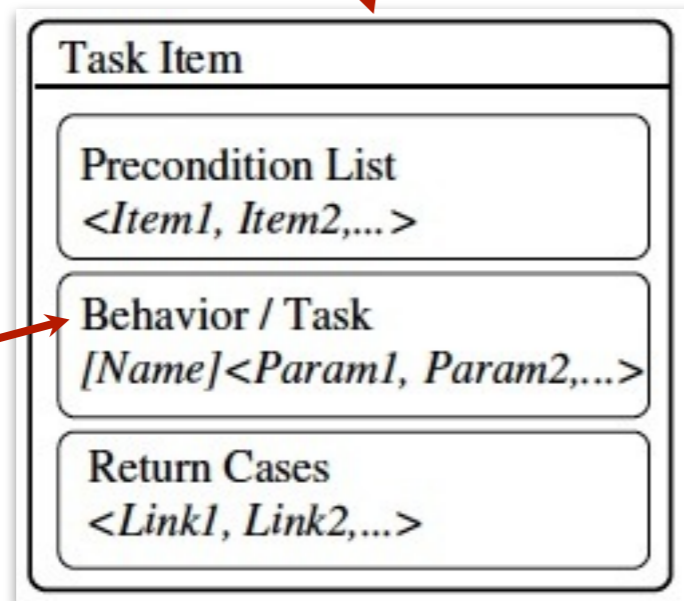
[Rybski et al. 2007]

# Learning a task plan: STRIPS-style plans

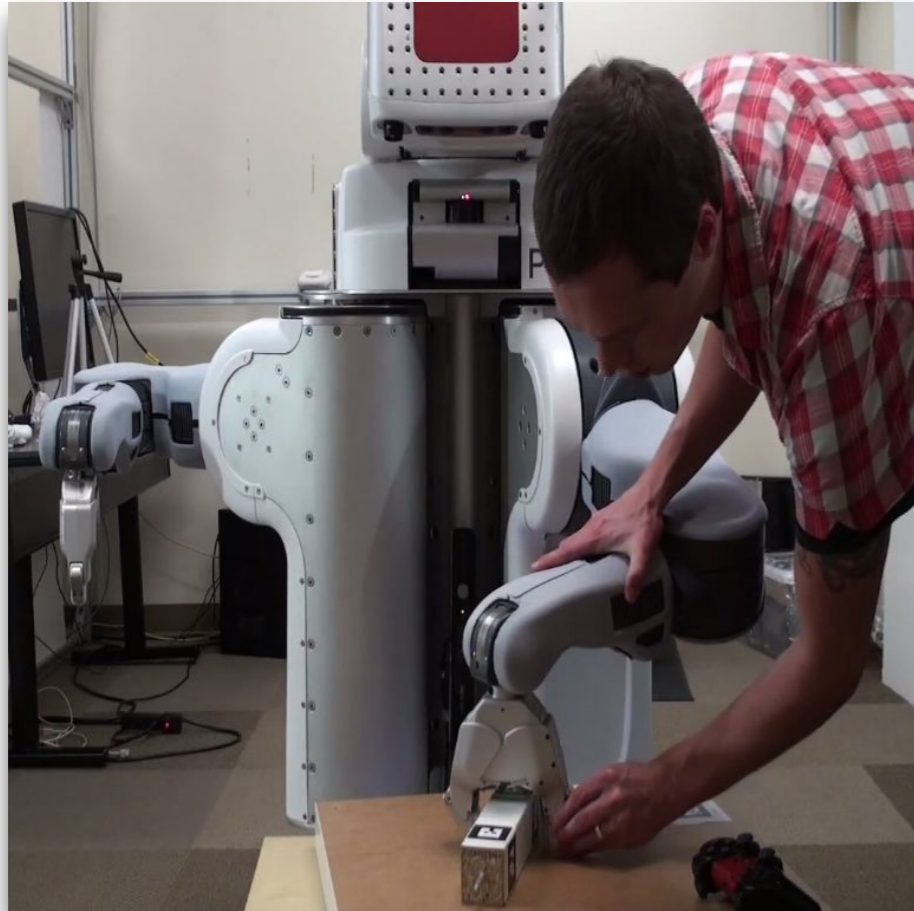
```

>> When I say deliver message
>> If Person1 is present
>>   Give message to Person1
>> Otherwise
>>   If Person2 is present
>>     Give message to Person2
>>   Otherwise
>>     Report message delivery failure
>> Before
>> Before
>> Goto home
  
```

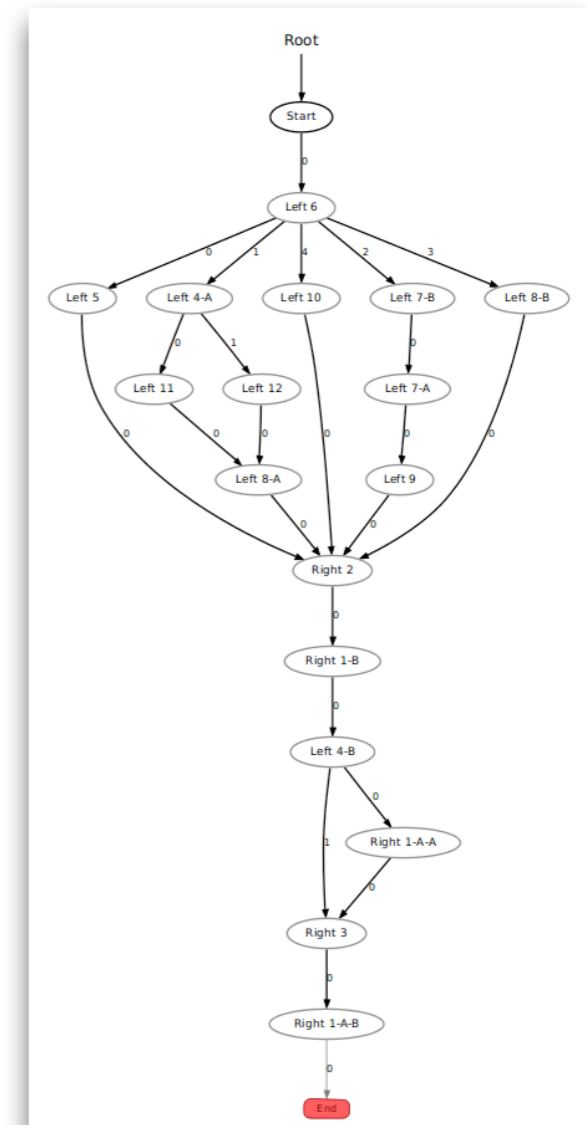
Demonstrated  
behavior



# Learning a task plan: Finite state automata



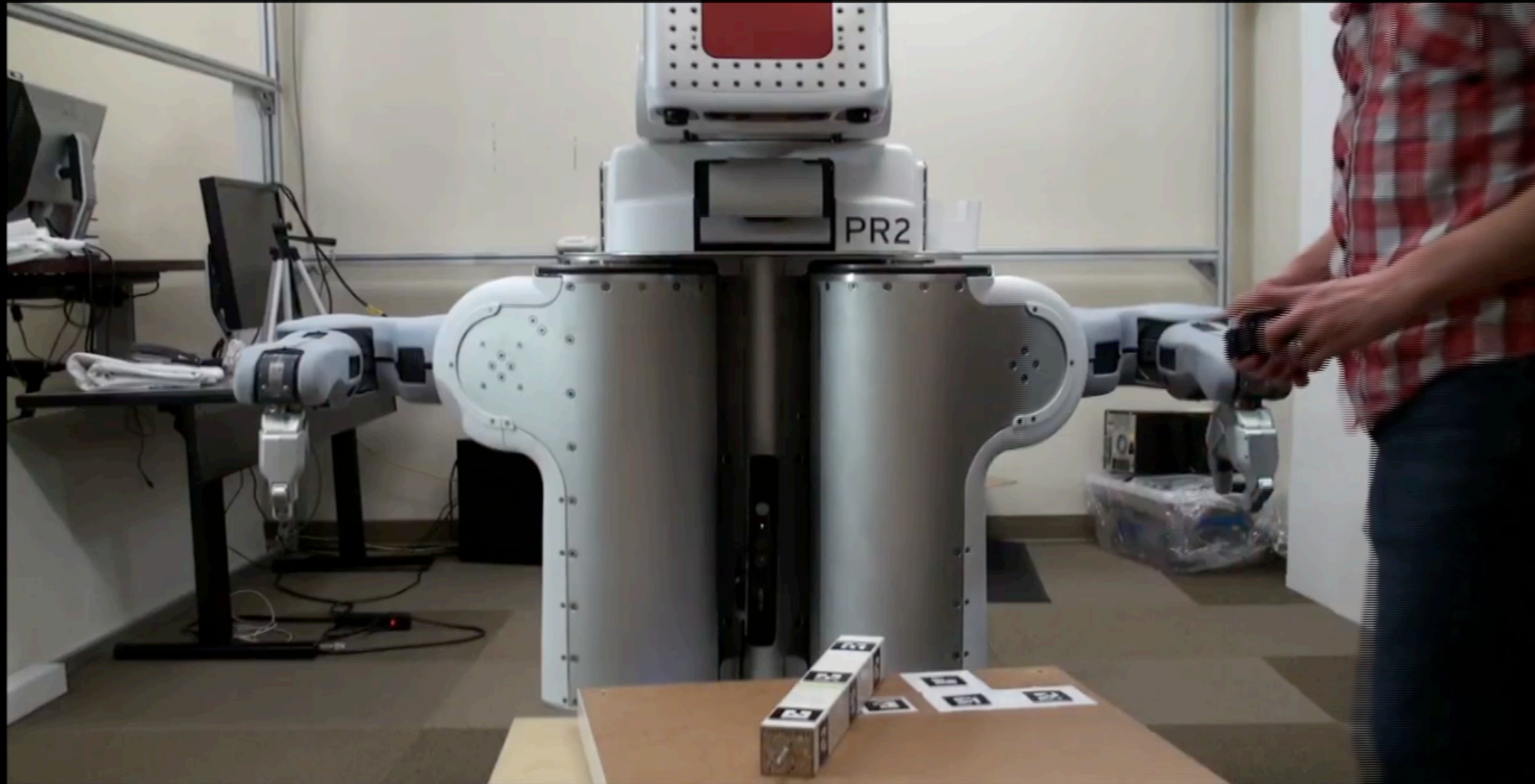
Unsegmented demonstrations  
of multi-step tasks



Finite-state task  
representation

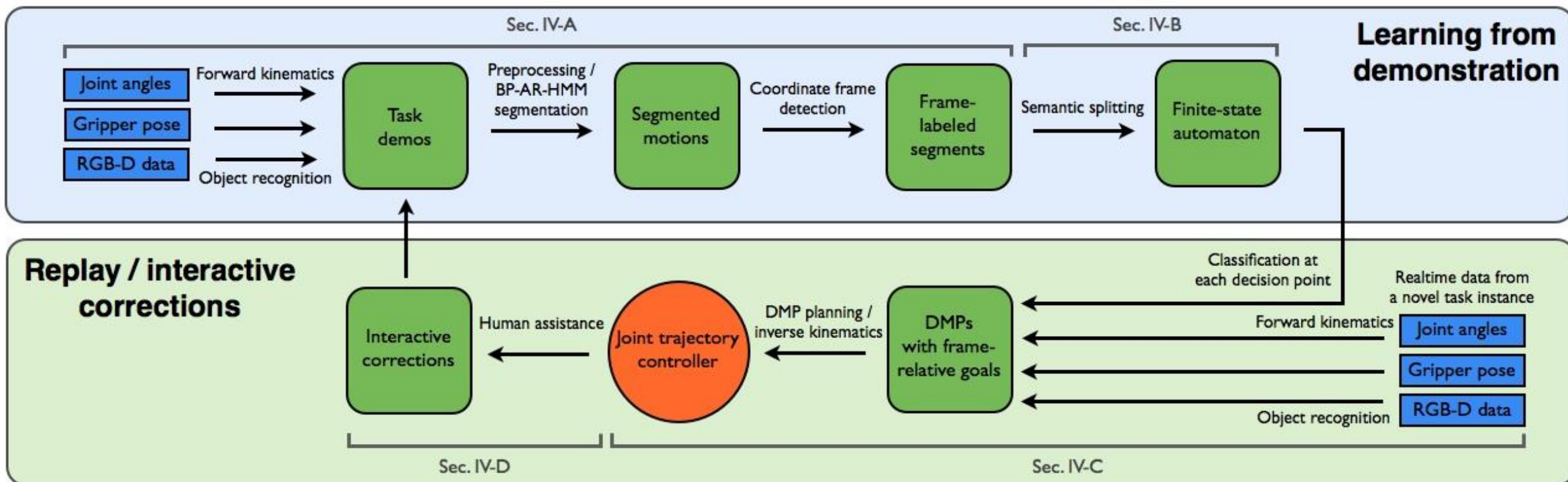
[Niekum et al. 2013]

# Learning a task plan: Finite state automata



4x

[Niekum et al. 2013]

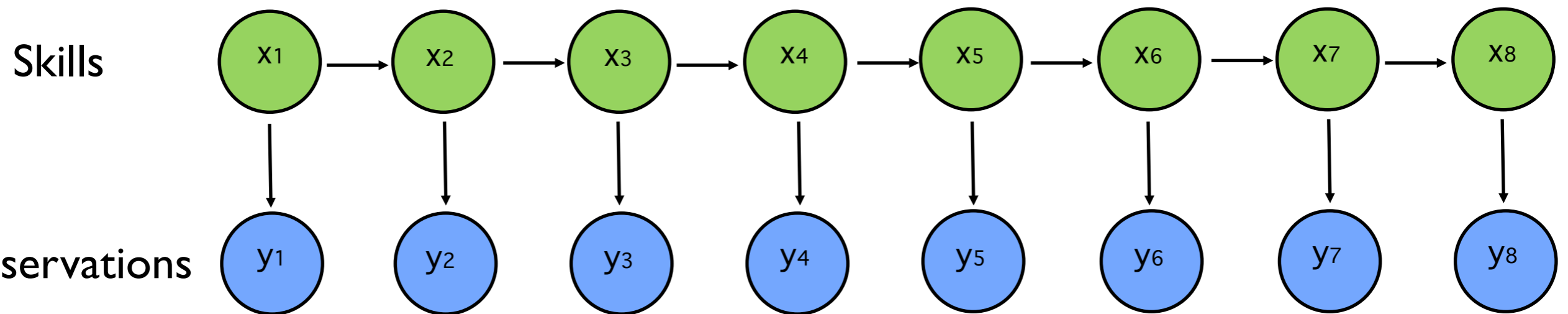


Ikea Assembly:

Overview of the iterative learning from demonstration framework



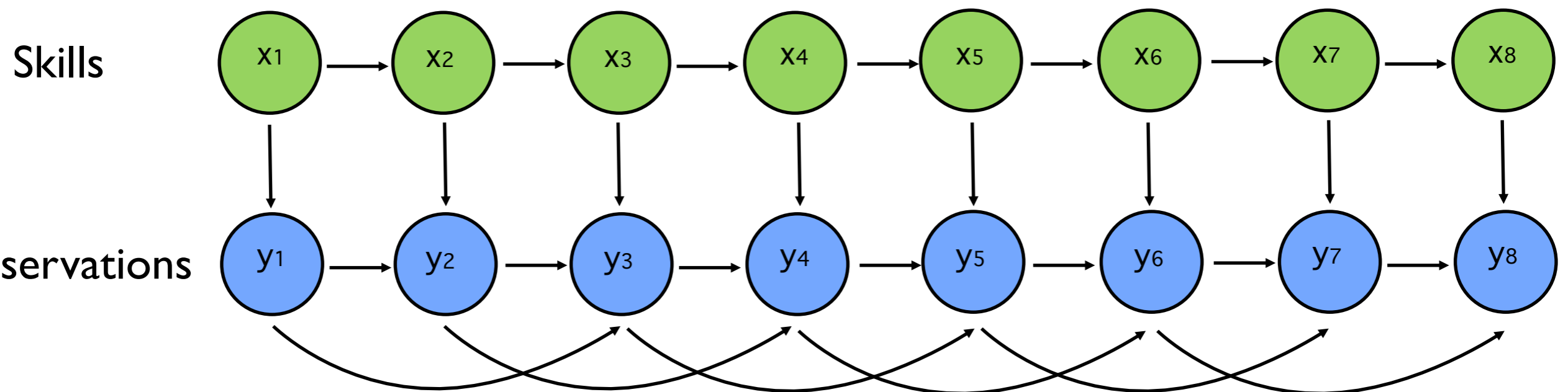
# Learning a task plan: Finite state automata



Standard Hidden Markov Model

# Learning a task plan: Finite state automata

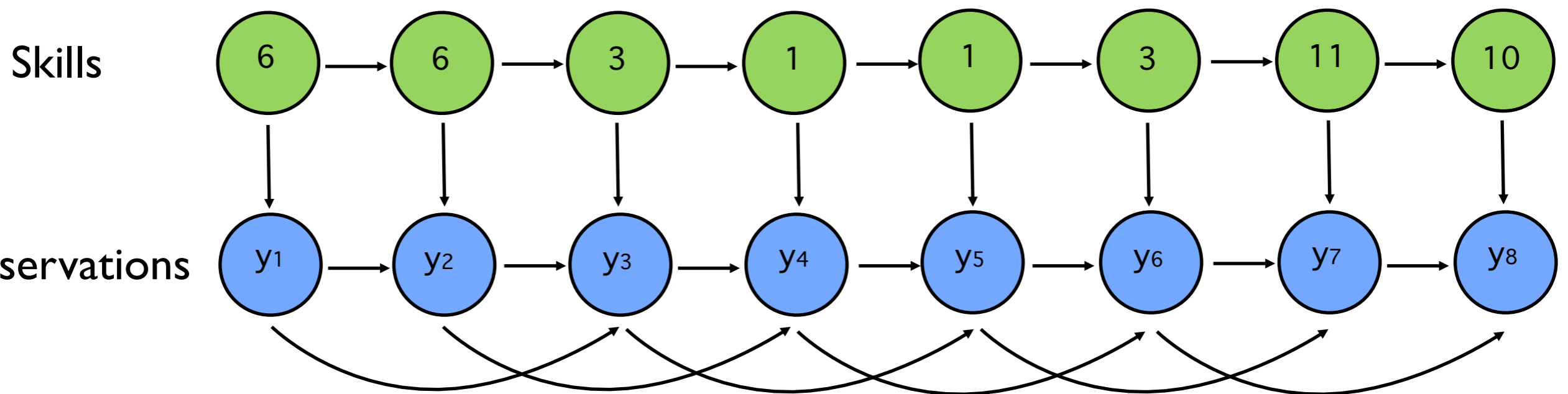
$$\mathbf{y}_t^{(i)} = \sum_{j=1}^r A_{j, z_t^{(i)}} \mathbf{y}_{t-j}^{(i)} + \mathbf{e}_t^{(i)}(z_t^{(i)})$$



Autoregressive Hidden Markov Model

# Learning a task plan: Finite state automata

$$\mathbf{y}_t^{(i)} = \sum_{j=1}^r A_{j, z_t^{(i)}} \mathbf{y}_{t-j}^{(i)} + \mathbf{e}_t^{(i)}(z_t^{(i)})$$

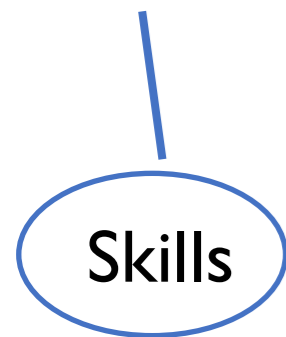


Autoregressive Hidden Markov Model

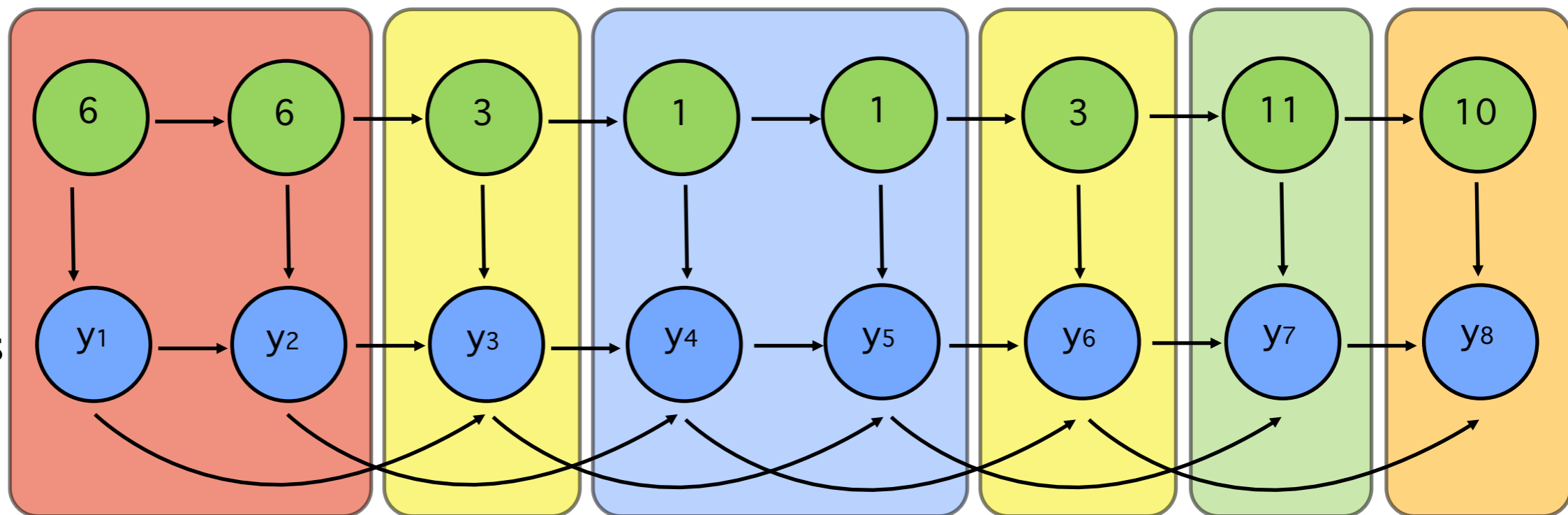
# Learning a task plan: Finite state automata

$$\mathbf{y}_t^{(i)} = \sum_{j=1}^r A_{j, z_t^{(i)}} \mathbf{y}_{t-j}^{(i)} + \mathbf{e}_t^{(i)}(z_t^{(i)})$$

unknown number!



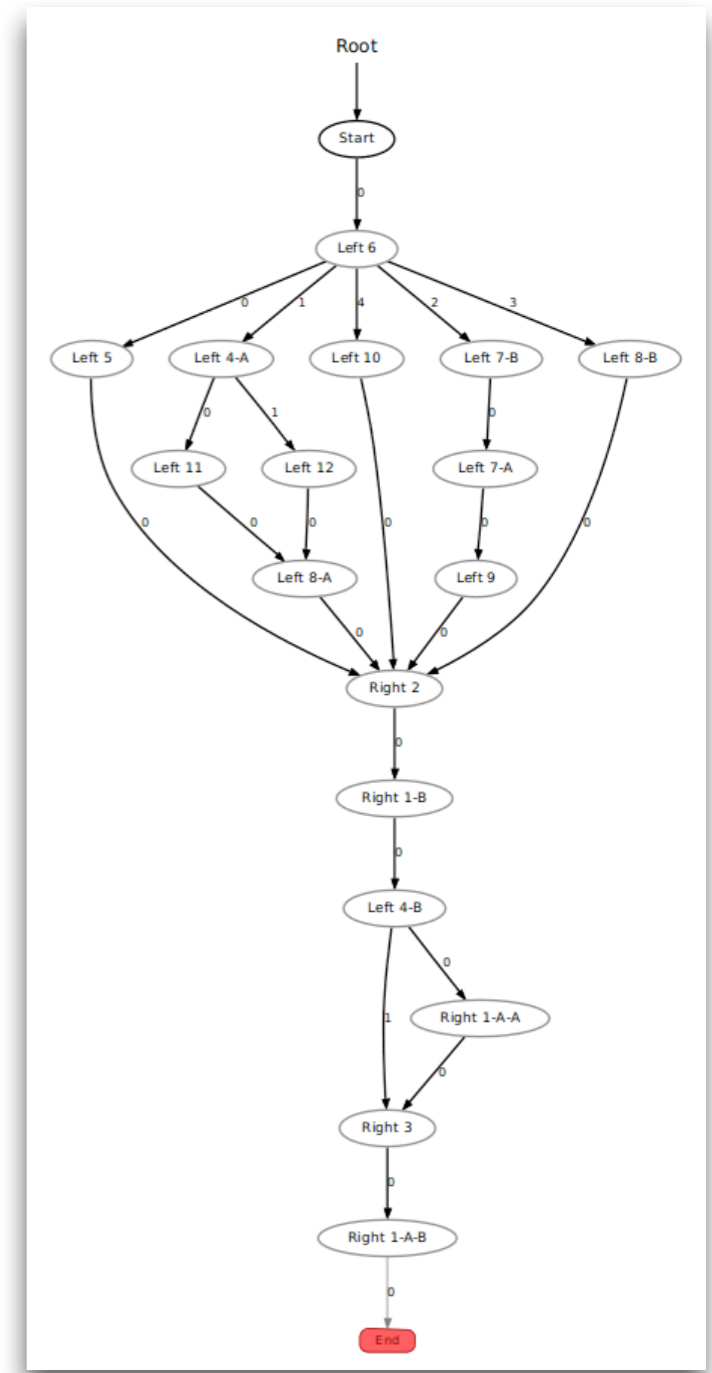
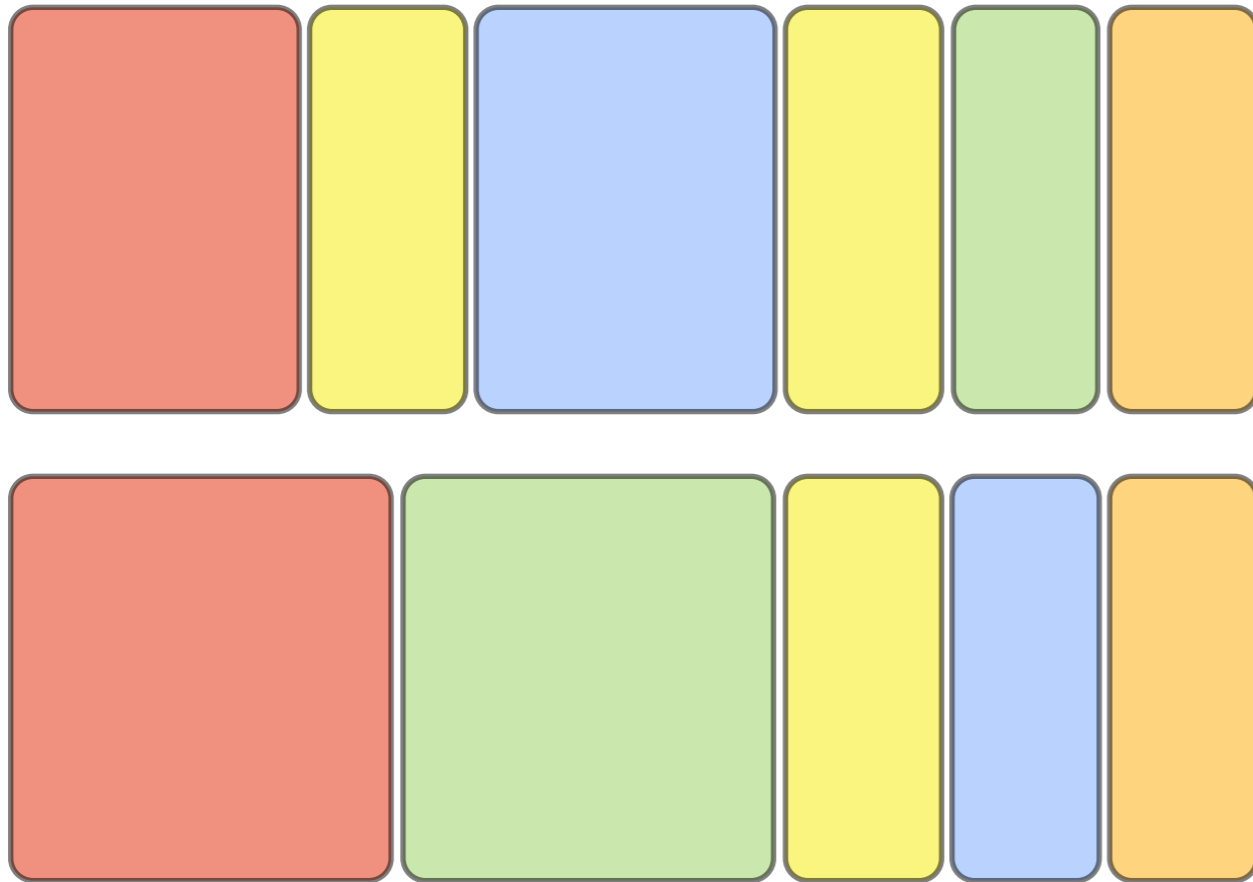
Observations



Beta Process Autoregressive Hidden Markov Model

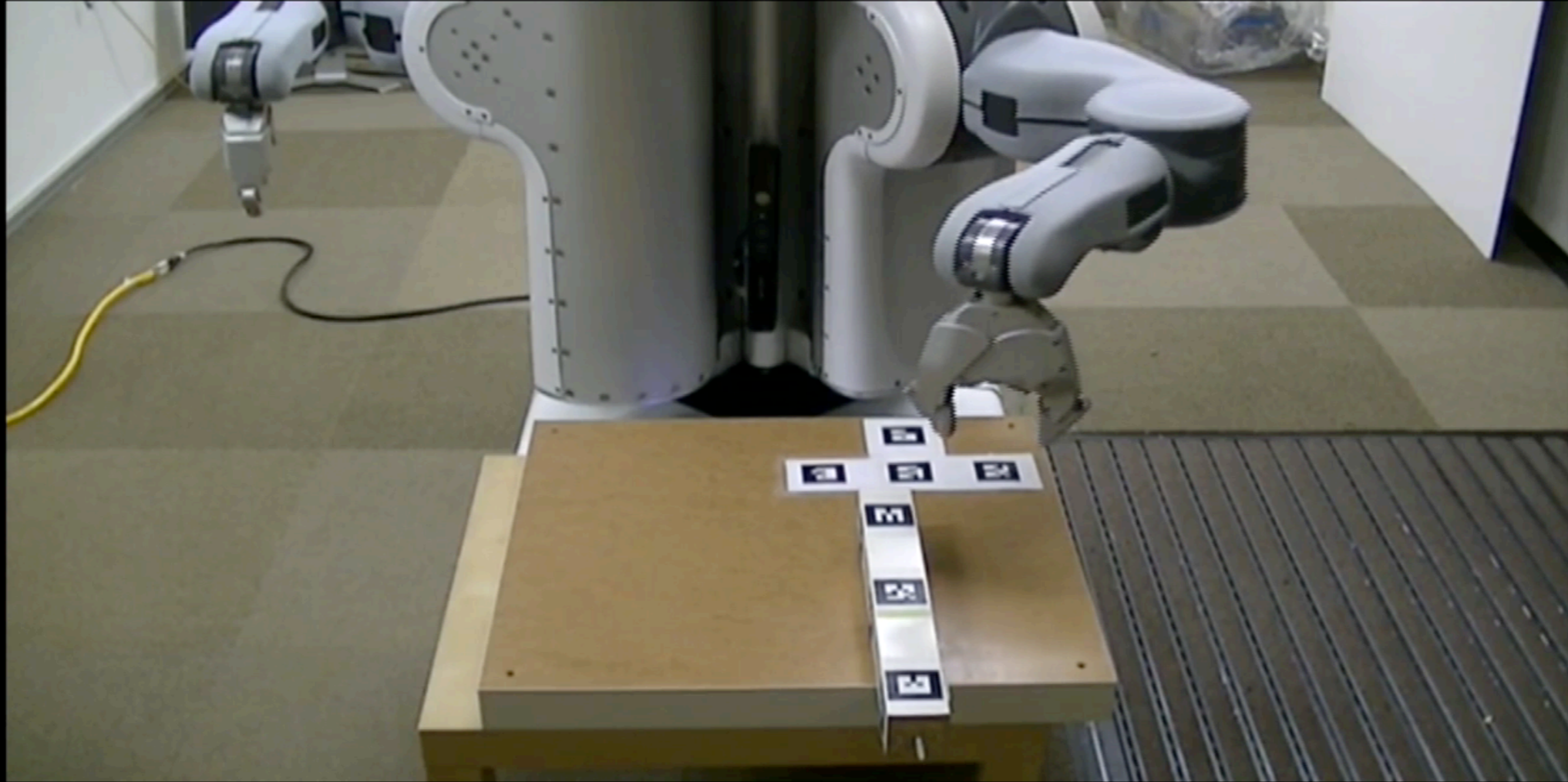
[Niekum et al. 2013]

# Learning a task plan: Finite state automata



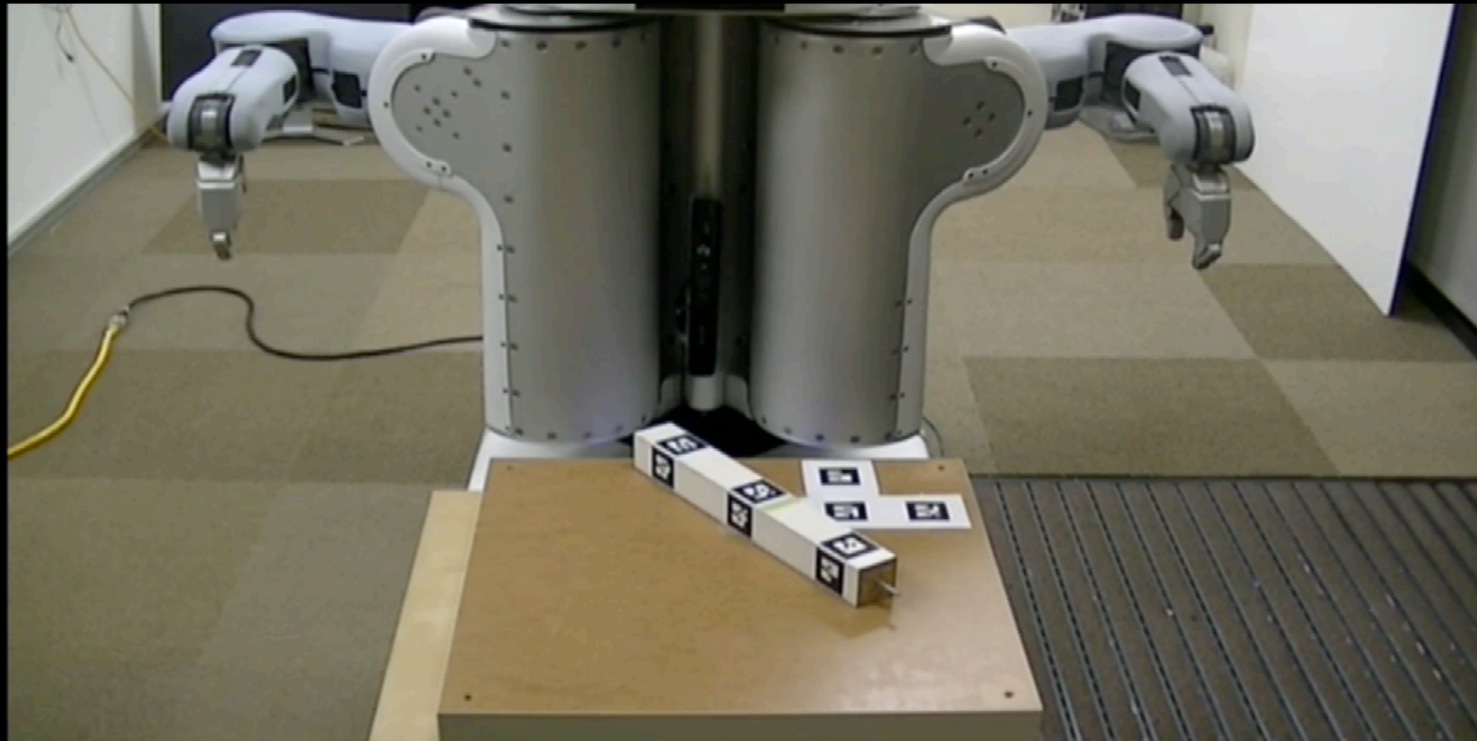
[Niekum et al. 2013]

# Interactive corrections

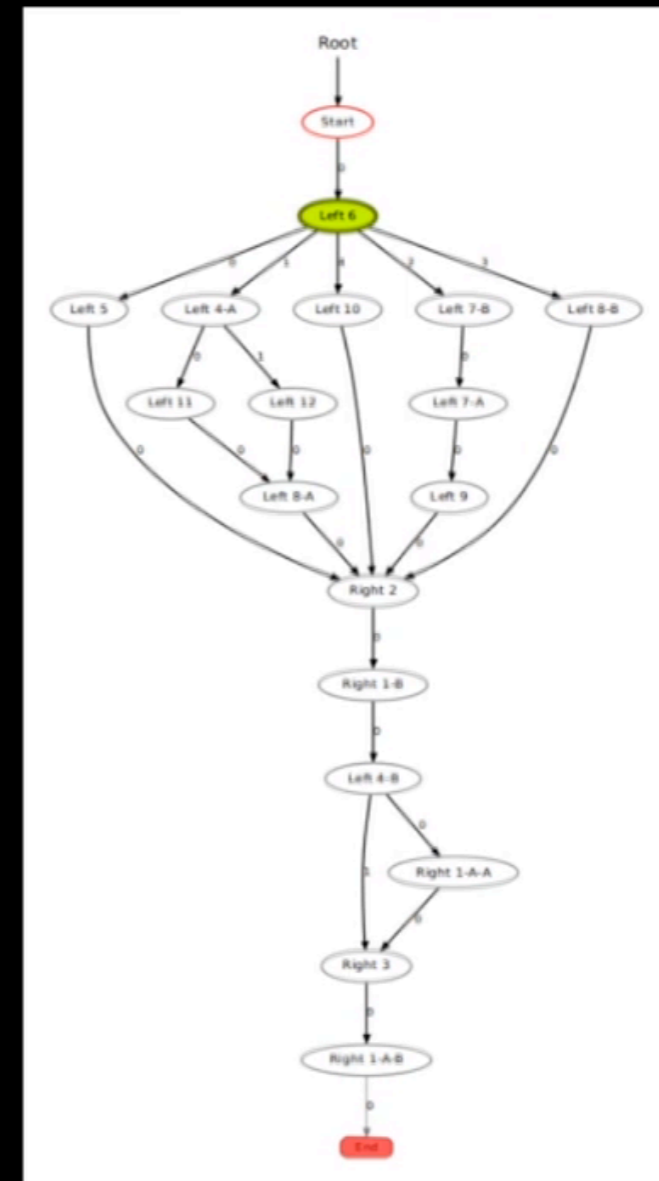


[Niekum et al. 2013]

# Replay with corrections: missed grasp

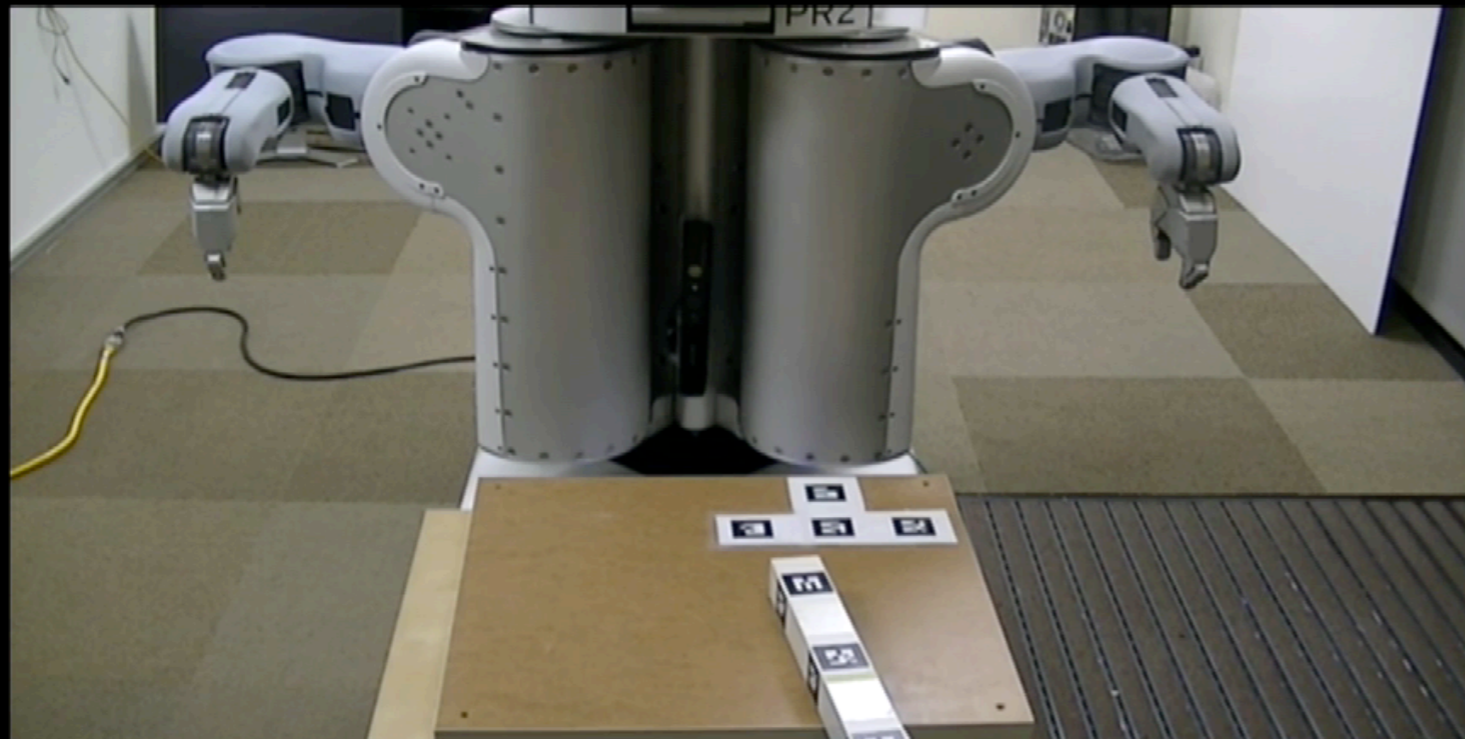


4x

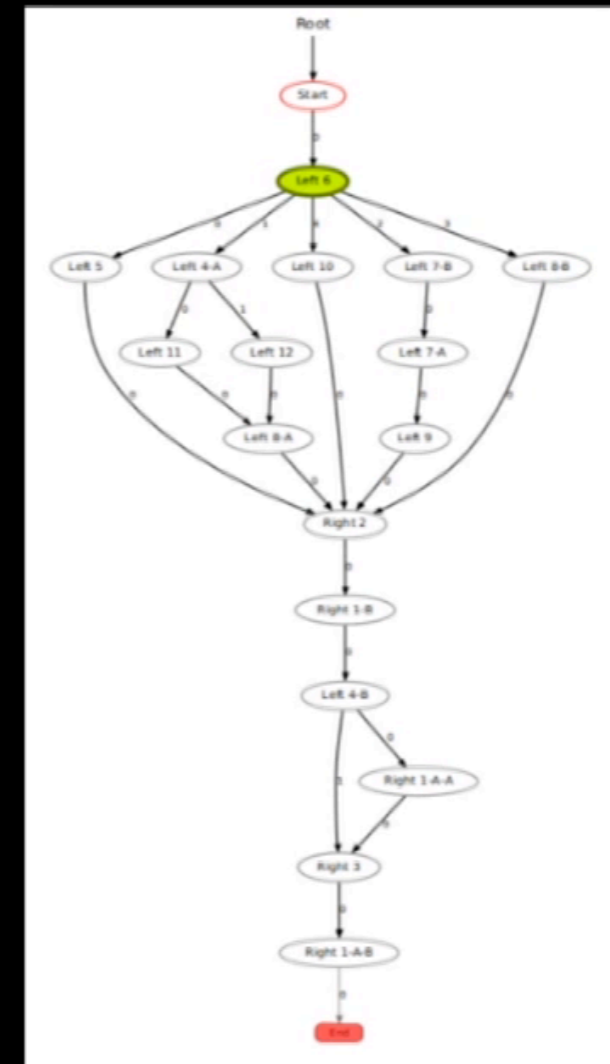


[Niekum et al. 2013]

# Replay with corrections: too far away



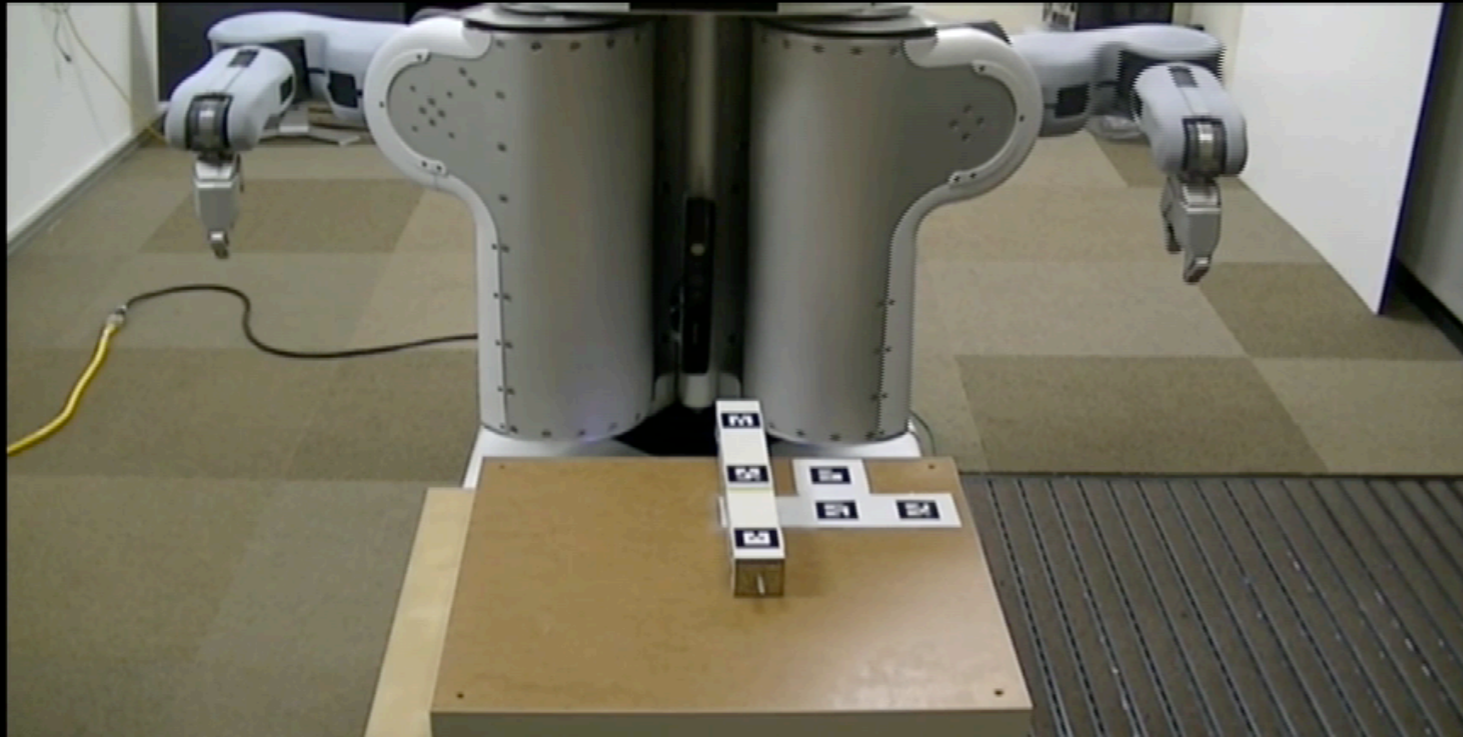
4x



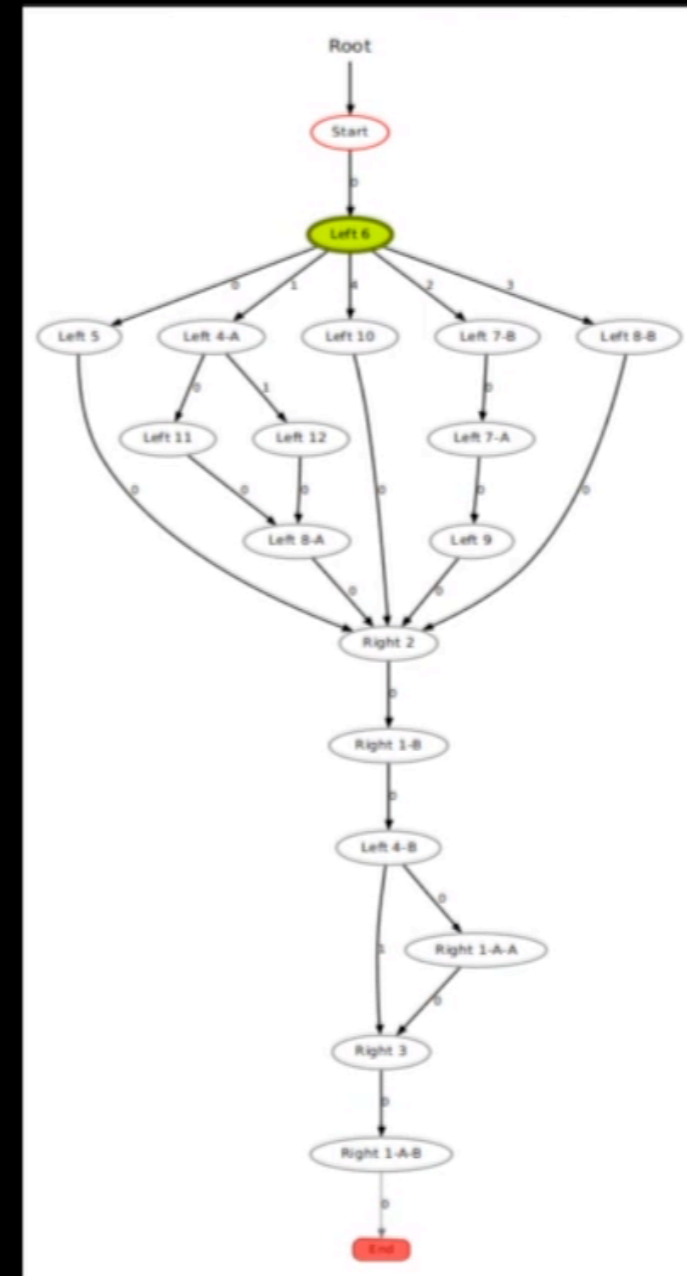
[Niekum et al. 2013]



# Replay with corrections: full run



4x

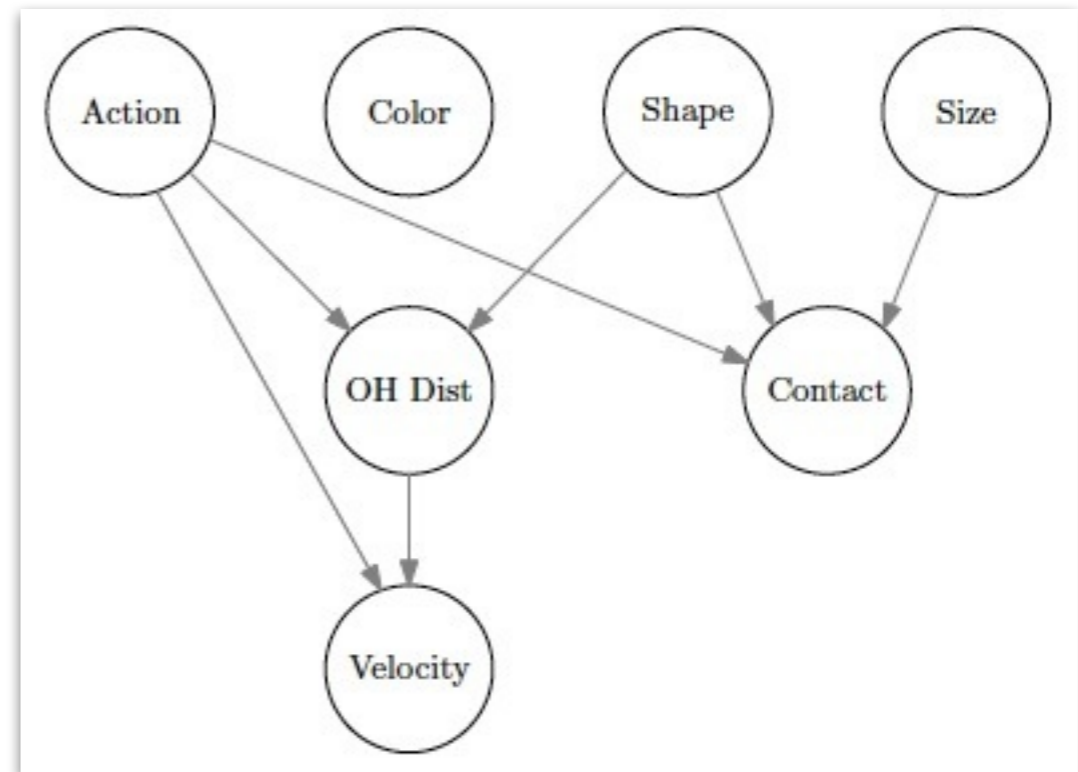


[Niekum et al. 2013]

# Learning object affordances: Action + object

Can we learn to recognize actions based on their effects on objects?

Random  
exploration



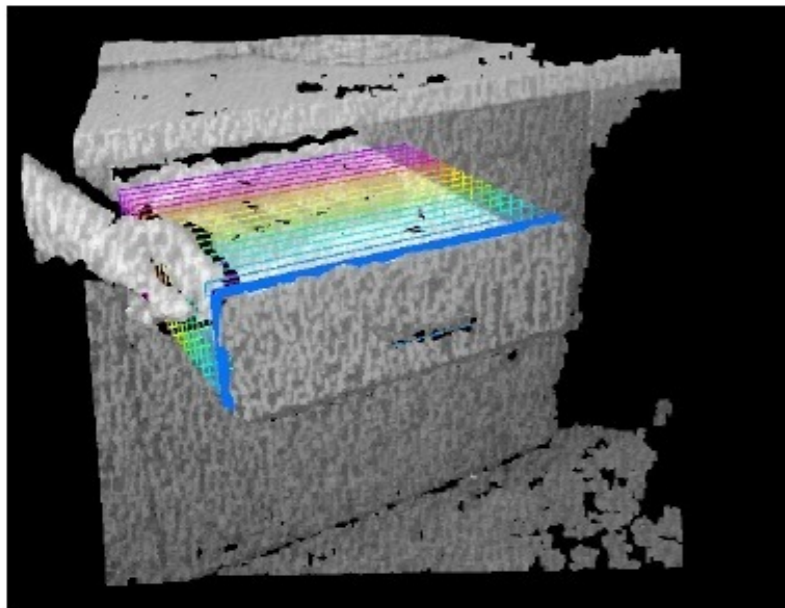
Object features: Color, shape, size

Actions: Grasp, tap, touch

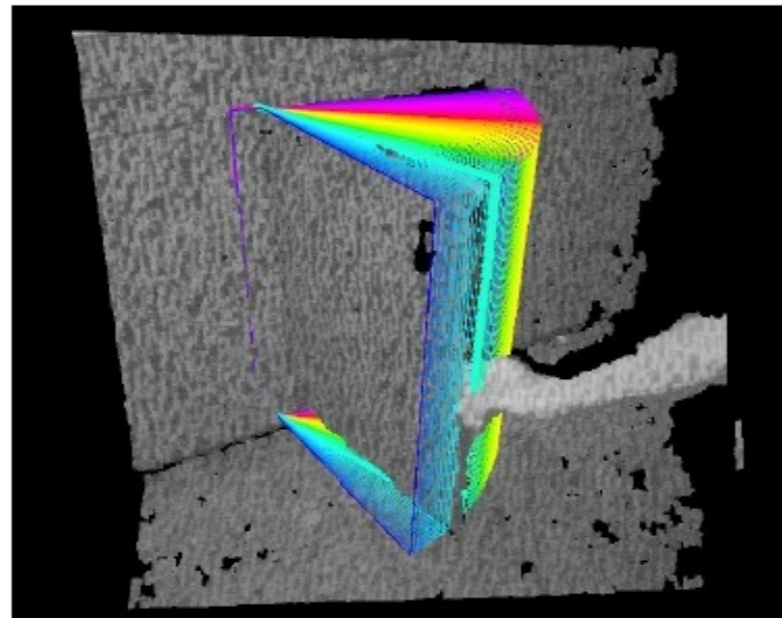
Effects: Velocity, contact, object-hand distance

[Lopes et al. 2007]

# Learning object affordances: Articulation models



Prismatic - drawer

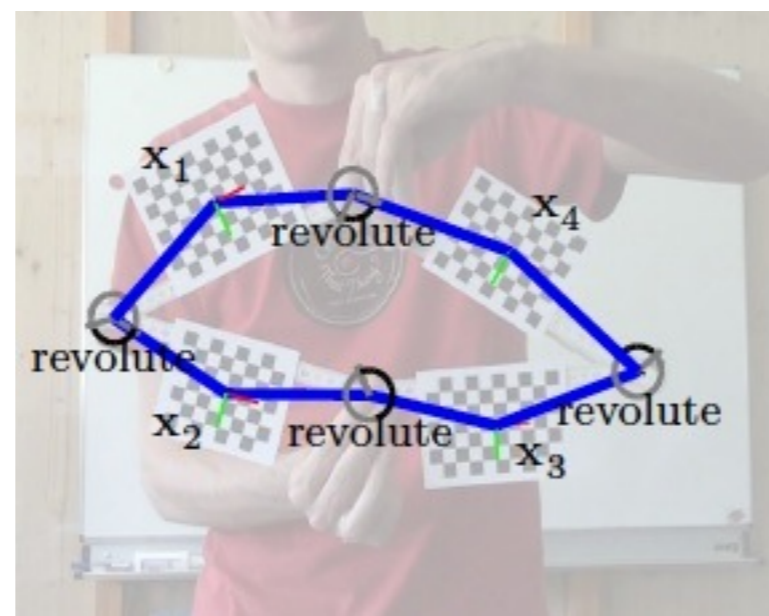


Revolute - cabinet



Gaussian process

-  
garage door



Infer full kinematic chain  
via Bayes net

[Sturm et al. 2011]

# Combining LfD with Reinforcement Learning

- Use human demonstrations to initialize the parameters of the controller.
- One cannot use directly demonstration as the dynamics of robot differ from human dynamics
- Use reinforcement learning to search for solutions nearby the demonstrations.



# Refinement through verbal interaction

- Robot has initial set of reaching skills
- Robot provided with a dialogue system to query the teacher
- Teacher modifies the controller through verbal guidance



Cakmak & Thomaz, Intern. Conf. on Human-Robot Interaction, 2012

# Future directions

- Feature selection
  - selecting too many features is computationally expensive and can “confuse” learning process, while too few features might lead to insufficient data for policy inference
  - What is an intuitive way to select the right features?
- Including temporal data
  - Currently, most algorithms discard temporal data
  - Repetitive tasks become difficult to sequentialize
  - Actions that have no perceivable effect on the states are difficult to learn from
  - Temporal data could alleviate both these issues

# Future directions

- Multi-robot demonstration learning
  - Both agents could request advice from human teacher or provide demonstrations for one another
- Refined evaluation metrics
  - Currently, LfD projects are highly domain and task specific
  - Field lacks a cross-domain standard for evaluating performance

# Future directions

- Multiple tasks, libraries of skills, skill hierarchies
- Parameterized skills (pick up any object, hit ball to any location, etc.)
- ‘Common sense’ understanding of physics, actions, etc.
- Bridge the gap between low-level observations and high-level concepts
- Novel ways to leverage human insight (natural language + demonstrations, learning to ‘play’, etc.)



P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. **An application of reinforcement learning to aerobatic helicopter flight.** In Neural Information Processing (NIPS'07), 2007.

P. Abbeel and A. Ng. **Apprenticeship learning via inverse reinforcement learning.** In Proceedings of the 21st International Conference on Machine Learning, 2004.

T. Cederborg, M. Li, A. Baranes, and P.-Y. Oudeyer. **Incremental local online gaussian mixture regression for imitation learning of multiple tasks.** In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010.

Luis C Cobo, Peng Zang, Charles L Isbell Jr, Andrea L Thomaz, and Charles L Isbell Jr. **Automatic state abstraction from demonstration.** In Twenty-Second International Joint Conference on Artificial Intelligence, 2009.

G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. **Robot learning from demonstration by constructing skill trees.** The International Journal of Robotics Research, 31(3):360–375, December 2011.

M. V. Lent and J. E. Laird. **Learning procedural knowledge through observation.** In K-CAP '01: Proceedings of the 1st International Conference on Knowledge Capture, 2001.

M. Lopes, F. S. Melo, and L. Montesano. **Affordance-based imitation learning in robots**. 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2007.

S. Niekum, S. Osentoski, C.G. Atkeson, A.G. Barto. **Online Bayesian Change-point Detection for Articulated Motion Models**. IEEE International Conference on Robotics and Automation (submitted), May 2015.

S. Niekum, S. Chitta, B. Marthi, S. Osentoski, and A. G Barto. **Incremental semantically grounded learning from demonstration**. In Robotics Science and Systems, 2013.

P. E Rybski, K. Yoon, J. Stolarz, and M. Veloso. **Interactive robot task training through dialog and demonstration**. In HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction, 2007.

M. Veloso, F. V. Hundelshausen, and P. E Rybski. **Learning visual object definitions by observing human activities**. In 5th IEEE-RAS International Conference on Humanoid Robots, 2005.

B. Akgun, M. Cakmak, J. Yoo, and A. L. Thomaz. **Trajectories and Keyframes for Kinesthetic Teaching: A Human-Robot Interaction Perspective.** In Proceedings of the International Conference on Human-Robot Interaction, 2012.

W. B. Knox and P. Stone. **Tamer: Training an agent manually via evaluative reinforcement.** In Proc. of the 7th IEEE International Conference on Development and Learning, 2008.

S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. **Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation.** AAAI, 2011.